# The International Congress for global Science and Technology


## ICGST International Journal on Artificial Intelligence and Machine Learning (AIML)

**Volume (16), Issue (I)**
**December, 2016**

**www.icgst.com**
**www.icgst-amc.com**
**www.icgst-ees.com**

AIML Journal
ISSN Print 1687-4846
ISSN Online 1687-4854
ISSN CD-ROM 1687-4862


# Table of Contents


**ICGST International Journal on Artificial Intelligence and Machine Learning (AIML)**

**A publication of the International Congress for global Science and Technology - (ICGST)**

**ICGST Editor in Chief**

**Dr. rer. nat. Ashraf Aboshosha**

**www.icgst.com, www.icgst-amc.com, www.icgst-ees.com**

**editor@icgst.com**

www.icgst.com

# Enhanced User Authentication through Keystroke Biometrics for Short-Text and Long-Text Inputs

M. AbdelDayem, H. Hemeda, A. Sarhan
*Department of Computer and Control Engineering, Faculty of Engineering, Tanta University, Tanta, Egypt*
*mahmoudabdeldayem@hotmail.com, hamed.hemeda@hotmail.com, amany_m_sarhan@yahoo.com*

## Abstract
Password typing is the most common authentication method to secure computer systems. However, due to its simplicity, it could be easily compromised. Keystroke dynamics can be combined with password checking to result in a more secure and robust system, inexpensively and effectively. This paper investigates the use of the keystroke dynamics biometric as an authentication method. We employ algorithms to authenticate users through their short-text and long-text typing patterns. The experimental results show that our system has high specificity when dealing with short-text input and high sensitivity when dealing with long-text input.

***Keywords:*** *Computer Security, Authentication, Biometrics, Keystroke Dynamics, Pattern Recognition, Identity Verification.*

**Nomenclature**

| | |
|---|---|
| FAR | False Accept Rate |
| FRR | False Reject Rate |
| EER | Equal Error Rate |
| P-R | Press-Release |
| P-P | Press-Press |
| R-P | Release-Press |
| VKF | Virtual Key Force |
| SVM | Support Vector Machine |
| NN | Neural Networks |
| WNN | Weightless Neural Networks |
| GA | Genetic Algorithms |
| PSO | Particle Swamp Optimization |
| XML | Extensible Markup Language |

## 1. Introduction
Computers are used to store and process sensitive and confidential information. It became necessary to secure this information from intruders. Authentication is the process of determining whether a user should be allowed access to a particular system or resource. Password typing is the most common authentication method for computer systems. However, it is vulnerable to imposter attacks. Therefore, other authentication methods, such as biometrics-based authentication methods, are used alone or integrated with password typing authentication.

Biometrics technologies provide promising means of identification and authentication. Biometrics is the science and technology of measuring and statistically analyzing biological data. Biometrics can be divided into two categories: physiological and behavioral [17]. Physiological characteristics refer to what the person is, such as fingerprint, face, iris, etc. Behavioral characteristics refer to what a person does, or how the person uses the body, such as voice, signature, keystroke dynamics, mouse dynamics, etc.

Keystroke dynamics is a process of analyzing the way a user types at a keyboard by monitoring it in order to identify the users based on habitual typing rhythm patterns. It is a very attractive method due to several reasons. Firstly, it is not intrusive and computer users frequently type on a computer keyboard. Secondly, it is inexpensive since the only hardware required is a computer and a keyboard. Thirdly, after an authentication phase has verified a user's identity, the user continues using the keyboard and that allows subsequent checking for the user's identity [5].

In this work, we propose a keystroke dynamics biometric system to verify the identity of users. We adopted different algorithms to authenticate users through their short-text and long-text data. We used a statistical method as our short-text algorithm and a k-nearest neighbor classification method as our long-text algorithm. The long-text algorithm depends on the relationship between users' data while the short-text algorithm depends on each user's data. We ran two experiments: the first one over a small set of eight users and the second one over a larger set of twenty six users.

The remainder of the paper is organized as follows: In the next section, we provide some general background information on keystroke dynamics authentication systems. Section (3) describes our proposed keystroke biometric system and the phases it constitutes. Section (4) describes our keystroke dynamics application, the experiments, presents the experimental results, and evaluates the performance of our system. Section (5) concludes the paper and provides suggestions for future work.

## 2. Related Work

Different features for classification were proposed by different studies. Latency between keystrokes is one of the most commonly used features by many researchers. There are three types of latencies as defined by Balagani et al. [4] - press-to-press (P-P), release-to-release (R-R) and release-to-press (R-P) latencies. The P-P latency is also defined as digraph and is used by most researchers. Trigraph is the time interval between the presses or releases of alternate keystrokes. Some researchers have also used other N-graph features for identification. The trigraph feature was first used by Bergadano et al. [6]. Key hold time or dwell time is defined as the time for which each keystroke was pressed and is used by many researchers as well. Robinson et al. [18] concluded that hold times are much more important than inter-key times. Certain types of keyboards are available in the market which can measure the pressure applied to a key while typing. Attempt to recognize emotion from users typing pattern using pressure sensitive keyboards was carried out by Lv et al. [15] A new feature called Virtual Key Force (VKF) was proposed by Shanmugapriya and Padmavathi [19]. It is calculated based on the typing speed and behavior of the user on the keyboard.

Different classification methods were used by different studies. Some studies used simple algorithms while others used more complex algorithms. Table 1 lists the different classification methods undertaken by researchers towards developing authentication and identification systems. The studies evaluate their systems' performances using False Accept Rate (FAR) and False Reject Rate (FRR) or Equal Error Rate (EER). FAR and FRR are described in Section 4.4. EER is the rate at which both accept and reject errors are equal.

We found that most previous keystroke biometric studies focused on short-text input, while a few studies focused on long-text input. No study has incorporated both short-text and long-text authentications before. The relationship between the two performance evaluation metrics (FAR and FRR; both are described in Section 4.4) is inversely proportional; when one decreases, the other increases. Thus, both metrics cannot be improved for an authentication system at the same time, as shown in table 1, without using sophisticated, computationally intensive algorithms. In this paper, we combine both short-text and long-text authentications and work on improving one performance metric for each authentication type while using simple algorithms. Another advantage that we gain from combining both authentications is that long-text authentication is very appropriate for dynamic authentication, where the user identity could be verified through his interaction with the secured system after he is authenticated, while short-text authentication is used for static authentication, where the user identity is verified during login. Moreover, we developed our own java application (see Section 4.1) and collected the data from our users instead of using readily available keystroke databases on the Internet [20, 27]. This gave us more flexibility in extracting the features and the information we need from our users to achieve the best performance for our system.

| Study | Classification Technique or Method | Results | | |
|---|---|---|---|---|
| | | FAR | FRR | EER |
| [10] | Distance measure | 8.33% | 3.33% | |
| [14] | Statistical | 4.5% | 5.5% | |
| [1] | Weight measure | 22.5% | 0% | |
| [2] | Neural Networks (NN) | 0% | 5.01% | |
| [22] | Weighted NN | | | |
| [20] | Parallel decision trees | 0.88% | 9.62% | |
| [7] | Gaussian Mixture Model | | | 1.3%<br>5.88% |
| [9] | Support Vector Machines (SVM) | | | 13.45% |
| [16] | Statistical | | | 12.7% |
| [3] | SVM with Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) | 0.43% | 4.75% | |
| [13] | Mean Nearest Neighbor-Multi Layer Perceptron (NN-MLP) | | | 9.96%<br>11%<br>16.24% |
| [5] | Random Forest | 1% | 14% | |

Table 1. Authentication and identification using different classification techniques.

## 3. Proposed Keystroke Biometric System

The proposed keystroke biometric system consists of the following phases: keystroke data collection and storage, feature extraction, text processing, and pattern classification and login phase. Figure 1 shows a flow diagram of the phases that form the proposed keystroke biometric system. The next subsections will describe each phase.

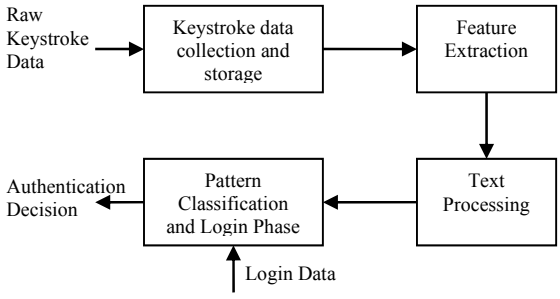



Figure 1. Flow diagram for the phases of the proposed keystroke biometric system.

latencies (described in Section 3.2.1) for both short-text and long-text inputs, but processed them differently in the text processing phase, to extract short-text and long-

text features. We modified the long-text hierarchy tree for keys durations (described in Section 3.2.2) by modifying the non-letters feature branch. We added more features for keys ratios for long-text authentication (described in Section 3.2.2). To improve the performance of our system, we added a parameters optimization phase for both short-text and long-text processing (see Sections 3.3.1.2 and 3.3.2.3). The significance of keys durations and transitions features for short-text authentication and keys durations, transitions and ratios features for long-text authentication were also tested to decide which features are the most appropriate for each authentication (see Sections 3.3.1.2, 3.3.2.3, and 4.3).

### *3.1. Keystroke data collection and storage phase*

The keys timing information is collected for each user for each key using key listeners in Java. This information is used to extract the features in the next phase. For every key pressed, the following information is collected: Key's character or name, Key's location on the keyboard (whether it is left, center, or right), Times when the key was pressed and released (in milliseconds).

### *3.2. Feature Extraction Phase*

In this phase, the required features are extracted from the raw keystroke timing information for both short-text and long-text inputs. The ways in which the short-text and long-text features are extracted are different. They are detailed in the next subsections.

#### *3.2.1. Short-text Features*

The following latencies [18] are calculated from the raw keystroke timing information and stored for further processing:

- Key Duration: It is the time interval between pressing a key and its release. It is sometimes called **Press-Release (P-R) latency**, **dwell**, or **hold time**.
- Key Transition, $T_1$: It is the time interval between pressing two successive keys. It is sometimes called **Press-Press (P-P)** or **digraph latency**.
- Key Transition, $T_2$: It is the time interval between releasing a key and pressing the next. It is sometimes called **Release-Press (R-P) latency** or **flight time**.

#### *3.2.2. Long-text Features*

The same latencies used for the short-text input are essentially used for the long-text input. They are computed for each character in the long-text data then the means and standard deviations for these latencies are calculated. They are then arranged in hierarchies for durations and transitions [21] as shown in Figures 2 and 3 respectively.

These hierarchy trees make use of the letter and digraph frequencies in English text. The left letters feature comprises the letters that are struck with the left hand (q, w, e, r, t, a, s, d, f, g, z, x, c, v, b) and the right letters feature comprises the letters that are struck with the right hand (y, u, i, o, p, h, j, k, l, n, m). The non-letters feature comprises two nodes: punctuation and numbers for the most frequent punctuation used in the English language and numbers, and modifiers for the frequent modifiers used in the English language such as shift, space, backspace, enter, etc.

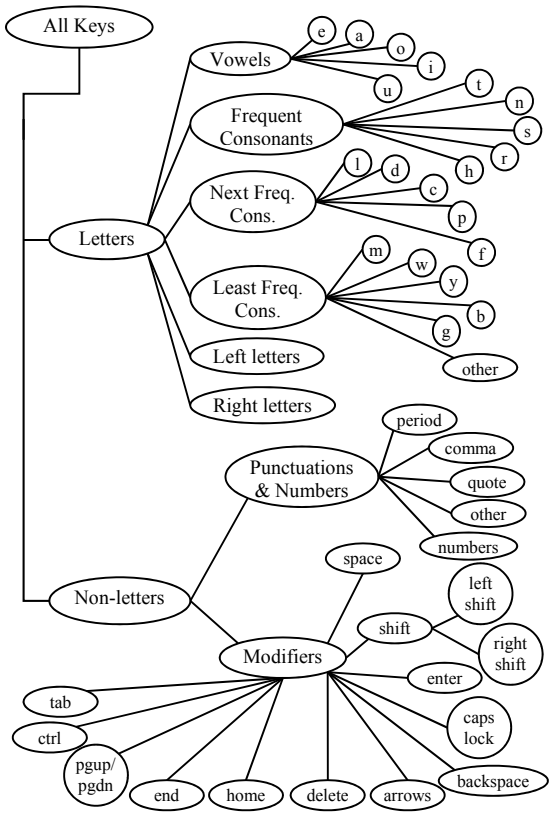



Figure 2. Hierarchy tree for all keys durations. Each category is represented by a mean and a standard deviation.

A feature vector, containing 251 features, is created for all means and standard deviations for durations (features 1-100), T1 (features 101-166), and T2 transitions (features 167-232). It also contains some useful ratios for key presses (features 233-246). These ratio features are useful to capture the users' preferences for using certain keys or key groups when they are typing, editing their text, and correcting their errors. The feature vector contains ratios of left mouse clicks, right mouse clicks, and double clicks (features 249-251). Finally, two other features are added as well (features 247-248). The first is the unadjusted keyboard input rate which equals the total time required to enter the text over the total number of key and mouse presses, and the second is the adjusted keyboard input rate which equals the total time required to enter the text minus any P-P time that is greater than 500 milliseconds (this represents the times the user is interrupted) over the total number of key and mouse presses.

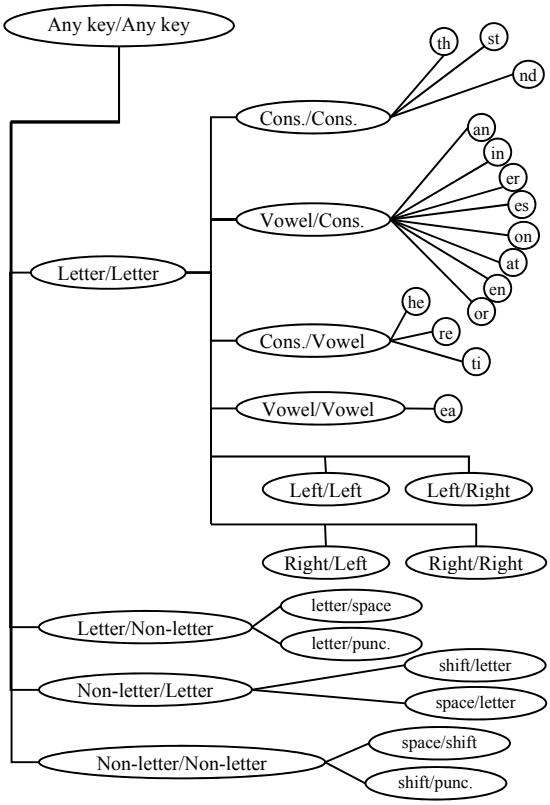



Figure 3. Hierarchy tree for all keys transitions [21]. Each category is represented by a mean and standard deviation for $T_1$ and $T_2$ transitions.

### 3.3. *Text Processing Phase*

Each user should enter his data many times to use them for training. Both short-text and long-text data are processed differently to prepare them for classification.

#### 3.3.1. *Short-text Processing*

Short-text processing is done in two steps: calculating means, standard deviations, and weights for letters' durations and transitions, and optimizing short-text parameters for classification.

#### 3.3.1.1. *Mean, Standard Deviation, and Weight calculation*

All samples for each user are compared with each other. Any redundant letter is removed. This will make all samples have the same length and letters.

The **mean**, $\mu_i$, and **standard deviation**, $\sigma_i$, for each letter duration and transition are computed. **Weights** are then computed for each duration and transition over all samples as follows:

$$weight_i = \left( \left(\frac{\mu_i}{\sigma_i}\right) \times \frac{1}{\sum_{j=1}^{n} \left(\frac{\mu_j}{\sigma_j}\right)} \right) \qquad (1)$$

It calculates the ratio of $\mu_i/\sigma_i$ for each duration or transition $i$ to the summation of all $\mu/\sigma$ for all durations or transitions, where $n$ is the number of samples. A short-text profile is created for each user containing mean, standard deviation, and weight for each letter duration and transition for both username and email. This profile is compared to login data supplied by the user during authentication and the system decides whether to accept login or not accordingly.

#### 3.3.1.2. *Parameters Optimization*

As the short-text authentication system is built, we create some constants. Instead of assuming specific values for these constants, it will produce much better results if we assume them as variables and optimize them in their reasonable ranges. We can get the best accuracy, sensitivity and specificity by optimizing these parameters. Moreover, using variables for the parameters will make our system more generic and can deal with any set of users as the parameters will be optimized according to the data the users supply.

The short-text parameters are optimized using a *hill climbing* method. Hill climbing [21] is a simple local search optimization technique. Since the optimized parameters have linear nature, hill climbing is very convenient to find their optimal values. Hill climbing works as follows: we assume a certain value for one of the parameters and run and test the system then, we change the parameter value and run and test the system while fixing the values of all the other parameters in the system. The value that produces the best performance for the system is considered the optimal value for the optimized parameter.

There are two types of testing to test the authentication system: legitimate data testing and imposter data testing. In legitimate data testing, the system is trained using the enrolled data and tested using a leave-one-out procedure on the enrolled data. The optimal parameter value should result in highest acceptance percentages in this type of testing implying few false rejects (Lowest FRR). In imposter data testing, the system is trained using the enrolled data and tested on imposter data supplied by the users logging in using each other's username/email. The optimal parameter value should result in lowest acceptance percentages in this type of testing implying few false accepts (Lowest FAR).

The short-text parameters that are optimized are $k_d$, $k_t$, $\tau_1$, $\tau_2$, $\%added_d$, $\%added_t$ and *acceptance%*. $k_d$ and $k_t$ are optimized in the range 1.0 to 4.0, in increments of 0.5, $\tau_1$ is optimized in the range 0.3 to 0.8, in increments of 0.05 and $\tau_2$ is optimized in the range 0.5 to 0.85, in increments of 0.05. $\%added_d$ and $\%added_t$ are optimized together in the ranges 0 to 1.0 and 1.0 to 0 in increments and decrements of 0.05 respectively so that their sum is always 1, and *acceptance%* is optimized in the range 0.2 to 0.8, in increments of 0.05. We have chosen these values for the optimization ranges because outside these ranges the authentication system will fail to work properly and will result in high percentages of false accepts or false rejects. For example, if we assumed the *acceptance%* equals 0.9, it means we need to have a 90% matching between login data and enrolled data to authenticate users which will result in many false rejects.

On the other hand, if we assumed the *acceptance%* equals 0.1, it means we have to have a 10% matching between login data and enrolled data to authenticate users which will result in high number of false accepts. This optimization procedure is repeated until the system produces the best results for both legitimate and imposter data testing.

The parameters cannot be optimized unless data by all users is supplied. Initial values of 1.0, 1.0, 0.65, 0.5, 0.5, 0.5 and 0.5 respectively are assigned to the parameters. We chose these initial values because these are median values that allow the system to perform well regardless of the set of users using it. We will describe the short-text parameters and their functions in Section 3.4.1.

#### 3.3.2. *Long-text Processing*

Many processing steps are carried out on long-text data to prepare it for classification. They are described in the next subsections.

##### 3.3.2.1. *Revised Mean and Standard Deviation Calculation*

If a key duration or transition exists in the long text very few times, it requires special handling when computing its mean and standard deviation. We make use of the parent nodes in the hierarchy trees for keys durations and transitions (see Figures 2 and 3) and use a fallback procedure to compute revised mean and standard deviation. This procedure is similar to the "*backoff*" procedures used in natural language processing [12].

So when the number of instances for a letter duration or transition is less than $\tau_{fallback}$, which is an optimized parameter, revised mean and standard deviation should be calculated [12]:

$$\mu' = \frac{n \times \mu + w_{fallback} \times \mu(fallback)}{n + w_{fallback}} \quad (2)$$

$$\sigma' = \frac{n \times \sigma + w_{fallback} \times \sigma(fallback)}{n + w_{fallback}} \quad (3)$$

Where $\mu'$ and $\sigma'$ are revised mean and standard deviation, $\mu$ and $\sigma$ are unrevised mean and standard deviation, $n$ is the number of instances, $\mu(fallback)$ and $\sigma(fallback)$ are fallback mean and standard deviation, and $w_{fallback}$ is another optimized parameter that converges the unrevised mean and standard deviation to the fallback mean and standard deviation in the fallback procedure. The fallback means and standard deviations are determined by the parent nodes in the hierarchy trees in Figures 2 and 3. For example, if the letter 'g' exists a few times (less than $\tau_{fallback}$) in the long text, its revised mean and standard deviation are calculated using the fallback mean and standard deviation of least frequent consonants, provided that they exist enough times in the long text.

##### 3.3.2.2. *Outlier Removal*

Outlier removal is very significant because a keyboard user could get interrupted while typing or pause for whatever reason, and the resulting outliers (usually overly long transition times) could skew the feature measurements. If any duration or transition is more than $k_\sigma$ standard deviations from their respective mean, it is considered an outlier and is removed. $k_\sigma$ is another optimized parameter. After all outliers are removed, means and standard deviations should be recalculated. Revised means and standard deviations should be recalculated, if needed. Outlier removal is performed *iterations* times, where *iterations* is an optimized parameter.

##### 3.3.2.3. *Parameters Optimization*

We assume the parameters created in our long-text authentication system as variables instead of constants and optimize them in their reasonable ranges for the same reasons stated in Section 3.3.1.2 for short-text data parameters. The long-text data parameters are optimized using a *hill climbing* method in the same manner as the short-text data parameters. The optimization process works in the same manner described in Section 3.3.1.2.

There are two types of testing to test the authentication system: legitimate data testing and imposter data testing. The system is trained using the enrolled data and tested using a leave-one-out procedure on the enrolled data for both types of testing. For legitimate data testing, the optimal parameter value should result in highest acceptance percentages implying few false rejects (Lowest FRR). For imposter data testing, the optimal parameter value should result in lowest acceptance percentages implying few false accepts (Lowest FAR).

The long-text parameters that are optimized are $\tau_{fallback}$, $w_{fallback}$, $k_\sigma$, *iterations*, *#neighbors*, $w_d$, $w_t$ and $w_r$. $\tau_{fallback}$ is optimized in the range 3 to 10, in increments of 1, $w_{fallback}$ is optimized in the range 1 to 5, in increments of 1, $k_\sigma$ is optimized in the range 1.0 to 4.0, in increments of 0.5, *iterations* is optimized in the range 0 to 4, in increments of 1, and *#neighbors* is optimized in the range 2 to 12, in increments of 2. $w_d$, $w_t$ and $w_r$ are optimized together so that their sum must equals 1.0. $w_t$ is optimized in the range 1.0 to 0, in decrements of 0.05, and $w_r$ is optimized in the range 0 to 0.3, in increments of 0.05, while $w_d$ will become $1.0 - (w_t + w_r)$.

The parameters cannot be optimized unless data by all users is supplied. Initial values of 5, 2, 1.0, 1, 10, 0.3, 0.55 and 0.15 respectively are assigned to the parameters. $\tau_{fallback}$, $w_{fallback}$, $k_\sigma$ and *iterations* are described in the previous sections (Secs. 3.3.2.1 and 3.3.2.2). We will


describe the remaining long-text parameters and their functions in Section 3.4.2.2.

#### 3.3.2.4. *Features Normalization*

The features used have different units and ranges: the means could be positive or negative (in case of overlapping $T_2$ transition); the standard deviations are always positive and their absolute values are usually much less than the means; and the ratios are small fractions. Since we are using distance-based classification method, the significant differences stated above will have big influence on the distance. This will have a very undesirable effect on our system. To eliminate this and to give each measurement roughly equal weight, all the feature values should be normalized to the range 0.0 to 1.0. Means, standard deviations, and ratios are normalized using the following formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

Where $x'$ is the normalized value, $x$ is the value to normalize, and $x_{min}$ and $x_{max}$ are its smallest and largest values respectively over all samples from all users [8].

### *3.4. Pattern Classification and Login Phase*

In this phase, different classification techniques are used for short-text and long-text data, and login data is compared to decide whether the user is authenticated or not.

#### *3.4.1. Short-text Login Phase*

When a user tries to log in, he enters his username and email. Both are compared with his short-text profile. We take the letters' durations for the username as an example. Every letter's duration is compared with its corresponding one in the profile to determine whether it falls in the $\mu \pm k_d \sigma$ range, where $k_d$ is an optimized parameter that controls the strictness of the $\mu \pm \sigma$ range; the bigger the value of the parameter, the more loose the range will be. If the letter's duration falls in the $\mu \pm k_d \sigma$ range, the letter duration's weight is added to the total weight of the login try for durations and the number of matches for the login try is incremented by one. When the comparison is performed for all letters' durations of the user's username, the total weight is compared with $\tau_1$ and the number of matches divided by the total number of characters in the username is compared with $\tau_2$. $\tau_1$ is an optimized parameter that represents the minimum weight value to consider a matching between login and enrolled data while $\tau_2$ is an optimized parameter that represents the minimum accepted value for (matches/characters). If the total weight is greater than $\tau_1$ or the number of matches counted over the total characters is greater than $\tau_2$, the value $\%added_d$ is added to the login try acceptance value for the username.

The same comparison is then done for all letters' transitions for the user's username using the parameter $k_t$ and adding the value $\%added_t$ to the login try acceptance value for the username, where $k_t$ is an optimized parameter that controls the strictness of the $\mu \pm \sigma$ range for letters' transitions. $\%added_d$ and $\%added_t$ are optimized parameters that determine which of the durations and transitions features are more effective in short-text authentication. For instance, if $\%added_d$ equals 0.9 and $\%added_t$ equals 0.1, this means that letters' durations matching between login and enrolled data adds 90% to the login try acceptance value while letters' transitions matching between login and enrolled data adds only 10% to the login try acceptance value.

The same comparison is then held for all letters' durations and transitions for the user's email yielding a login trial acceptance value for the email. The login trial acceptance value is the average of both username and email values. The login trial is considered successful if the login trial acceptance value is greater than or equal to *acceptance%*, which is an optimized constant that represents the minimum acceptance percentage to deem the login trial acceptable.

#### *3.4.2. Long-text Pattern Classification and Login Phase*

A Nearest Neighbor classifier [11] is used for verification in our system. Nearest neighbor classifier is a distance-based classifier that uses *Euclidean distance* between the feature vectors of test and training data to verify the identity of the user.

##### *3.4.2.1. Dichotomy Transformation Process*

Our system is not concerned with user identification. It is only concerned with user verification. The authentication problem is a multi-class (*polychotomy*) problem therefore; it will get much easier if we transformed it to a two-class (*dichotomy*) problem [21]. The transformation is done by calculating vector distances between pairs of samples of the *same* user (*intra-user distances*, denoted by $x_{intra}$) and pairs of samples of *different* users (*inter-user distances*, denoted by $x_{inter}$) as follows:

$$x_{intra} = (d_{ij}, d_{ik}) = |d_{ij} - d_{ik}|, i = 1 \; to \; n, \; and \; j, k = 1 \; to \; m, j \neq k \quad (5)$$

$$x_{inter} = \delta(d_{ij}, d_{kl}) = |d_{ij} - d_{kl}|, i, k = 1 \; to \; n, \; i \neq k, and \; j, l = 1 \; to \; m \quad (6)$$

Where *n* is the number of users, *m* is the number of samples, and $d_{xy}$ is the vector of values for all features of sample *y* supplied by user *x*. The *intra-user* and *inter-user distances* are calculated as the absolute differences between means and standard deviations for durations, transitions, and ratios of all letters in enrolled data.

#### *3.4.2.2. Long-text Login Phase*

When the user logs in, the *login intra-user distances* are calculated between the login data and corresponding enrolled data. Since we are using a distance-based classification method, Euclidean distances are calculated for durations, transitions and ratios between: *Login intra-user distances* and *Enrolled intra-user distances*, and *Login intra-user distances* and *Enrolled inter-user distances*.

Euclidean distance is the square root of the sum of squared distances. We use a k-nearest neighbor classifier. *#neighbors* is an optimized parameter that represents the number of neighbors required for classification. The Euclidean distances for durations, transitions and ratios are compared together and according to *#neighbors* the matching percentages are calculated for durations, transitions and ratios. To calculate the login trial acceptance percentage for the long-text, the matching percentages are multiplied by the constants $w_d$, $w_t$ and $w_r$ respectively as follows:

$$acceptance\% = w_d * matching\%_d + w_t * matching\%_t + w_r * matching\%_r \quad (7)$$

Where $matching\%_d$, $matching\%_t$ and $matching\%_r$ are the matching percentages calculated for durations, transitions and ratios, *acceptance%* is the login trial acceptance percentage, and $w_d$, $w_t$ and $w_r$ are optimized parameters that represent weights for durations, transitions and ratios matching percentages. These weights determine which of the durations, transitions and ratios features are more effective in long-text authentication. For instance, if $w_d$ equals 0.7, $w_t$ equals 0.3 and $w_r$ equals 0.0 (remember that the sum of the three parameters must equals 1.0) this means that *acceptance%* now equals ($0.7 \times matching\%_d + 0.3 \times matching\%_t$) giving a 70% weight for durations features, 30% weight for transitions features and no weight to ratios features. The login trial is considered successful if the login trial acceptance percentage is greater than or equal to 50%.

## 4. Experiments

To verify the effectiveness of the proposed system, two sets of experiments were carried out; the first time was carried out for eight users and the second time was carried out for twenty six users (including the first eight users). Each user supplied five samples for training and two samples for testing. Some users supplied samples for imposter testing.

### *4.1. Keystroke Dynamics Application*

We developed a Java application (see Figure 4) to collect the keystroke data. The users are required to type in their name and email in their respective fields. We first required the users to type to type username/password combinations but found it difficult to make users type their real passwords in our application for confidentiality reasons, and it will deteriorate the reliability of our system if they typed dummy passwords they are not used to. The username and email that users type should be at least 20 characters combined. The username/email combination forms the short-text input. The user is required to copy a predefined long-text [21] for training (see Text 1 in Appendix) into a text field. It contains 127 words (632 characters). For testing, the users either copied the same long-text used in training or another text [21] that has rather similar components (see Text 2 in Appendix). This text contains 107 words (556 characters).

To help the users keep track of the number of times they entered their data to the application, we used a submission number, as shown in Figure 4, which is automatically incremented according to the number of samples they have submitted before. The users should enter short-text data without errors or pause.

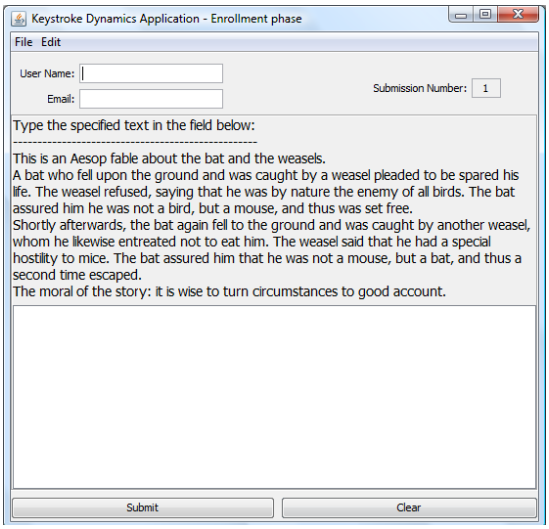


Figure 4. Keystroke Dynamics Application

Users are not allowed to make errors when enrolling their short-text data. Errors are allowed during the login phase only but not during data enrolment. Users are allowed to type in their names and emails sequentially, therefore backspaces are allowed, but not inserting letters in the middle of the words. The user is free to type long-text data in his normal typing pattern. He can delete, edit, and do whatever suits him. Whenever a user is not comfortable with his typing, or had an error during enrolling his short-text data, he can clear the data, by

pressing the clear button, and starts typing again. When the user finishes typing, he presses the submit button. When the data is submitted, it is stored in three XML files. The first file contains the raw keystroke timing information for username, email, and long-text data, the second file contains the short-text latencies for the username and email, and the third file contains the features for long-text data.

### 4.2. *Data Collection and Environmental Conditions*

We collected the keystroke data from the volunteers instead of using available keystroke data on the internet. Users were asked to enter their data five times. They were asked to leave at least 12 hours between entering training samples, so that a user does not adopt a certain typing pattern that he can't apply afterwards when logging in. Also, the samples need to be spread out over time similar to what might be expected in an actual application environment.

All users used laptops for entering the data. They used the same keyboard during training and testing. Twenty three users used their personal computer and keyboard for entering the data while three users didn't use their personal computers and keyboards. Also, the users were asked to be relaxed when typing their samples and so they were given the java application to allow them to type at home when they are in good mood.

Table 2 shows some information about the users who participated in the experiments. We represent the users with their initials (first column in the table). The first eight users participated in both experiments while the remaining users participated in the second experiment only. The typing speed is estimated for the average time the users took to enter their long-text data. The error rate is estimated for the average number of characters typed by the users to correct their errors (backspace, arrows and delete keys). Typing speed and error rate values are displayed in the table in brackets in their respective columns. Note that the levels for typing speed and error rate in the table are estimated for the users with respect to each other.

### *4.3. Experimentation and Optimization Results*

In this section, we describe the two experiments and present the optimization results for each.

#### 4.3.1. First *Experiment (8 users)*

All the eight users entered Texts 1 and 2 for testing (see Appendix). Six users were asked to try to log in using the other users' username/email and Text 1 (the same long-text used by all users in training) to provide imposter data (48 samples). They were asked to type normally. They didn't try to employ other users' typing pattern.

The optimization process for short-text and long-text data parameters for eight users resulted in the values shown in Table 3. For short-text data parameters, the values for $\%added_d$ and $\%added_t$ of 0.0 and 1.0 respectively mean that we relied **only** on keys transitions features for short-text authentication for eight users and keys durations features were redundant. The value for *acceptance%* of 0.55 means that the system needs 55% matching between login data and enrolled data to consider the authentication successful for this set of users.

| User | Gender | Age | PC Brand and Model | PC? | Typing Speed | Error Rate |
|---|---|---|---|---|---|---|
| AM | Male | 29 | Macbook Pro 2012 | Yes | Fast (2'51") | Very High (78) |
| AS | Male | 21 | Toshiba Satellite A350-13A | No | Medium (4'27") | High (36) |
| MA | Male | 28 | HP Pavillion G Series G6 584032-001 | No | Medium (5'55") | Medium (22) |
| KD | Male | 29 | Toshiba Satellite A100-813 | Yes | Medium (4'18") | Very Low (6) |
| KK | Male | 28 | HP Pavillion G Series G6-1162ex | Yes | Medium (5'40") | Very Low (8) |
| MD | Male | 27 | Toshiba Satellite A350-13A | Yes | Fast (3'19") | Very High (73) |
| MS | Male | 19 | Toshiba Satellite A350-13A | No | Slow (6'58") | High (43) |
| MG | Male | 27 | HP Pavillion G Series G6-1236ee | Yes | Medium (5'36") | Medium (25) |
| AB | Male | 25 | Dell Inspiron 15R (N5110) | Yes | Slow (7'11") | Medium (27) |
| AY | Female | 48 | Acer | Yes | Medium (4'57") | Medium (27) |
| AA | Male | 21 | Dell Inspiron 15 | Yes | Medium (5'30") | Low (11) |
| BG | Female | 28 | Dell Inspiron 15 | Yes | Slow (6'39") | Medium (26) |
| DM | Female | 27 | HP | Yes | Slow (7'43") | Medium (21) |
| FA | Female | 25 | HP Pavilion G series G6 | Yes | Medium (5'56") | Low (17) |
| GK | Male | 25 | Lenovo z510 | Yes | Fast (2'44") | High (36) |
| GS | Female | 21 | HP Pavilion G Series G6 | Yes | Very Slow (14'54") | Medium (27) |
| KH | Female | 21 | HP Pavilion G Series G6 | Yes | Slow (8'53") | Very High (91) |
| ML | Female | 26 | Dell Inspiron N5010 | Yes | Medium (4'59") | Low (12) |
| KO | Male | 21 | HP | Yes | Fast (3'22") | Medium (22) |
| MB | Female | 28 | Dell Inspiron 15-3537 | Yes | Slow (6'57") | Low (12) |
| ME | Female | 23 | Fujitsu AH530 Lifebook A Series | Yes | Very Slow (12'25") | Low (11) |
| SL | Male | 21 | Fujitsu Lifebook S760 | Yes | Fast (3'30") | High (39) |
| RS | Female | 25 | Lenovo SL500 Thinkpad | Yes | Medium (4'10") | Very Low (6) |
| SD | Male | 21 | Dell Inspiron 15 3521 | Yes | Slow (7'2") | Medium (24) |
| TA | Male | 23 | MSI CR620 | Yes | Very Fast (1'58") | Low (12) |
| TY | Female | 21 | HP EliteBook 8440p | Yes | Slow (6'36") | High (45) |

Table 2. Information about the users who participated in the experiments.

For long-text data parameters, the value for *iterations* of 0 means that the outlier removal process in the long-text processing phase was not required for this set of users. The values for $w_d$, $w_t$ and $w_r$ of 1.0, 0.0 and 0.0 respectively mean that we relied **only** on keys durations features for long-text authentication for eight users and keys transitions and ratios features were redundant.


| Parameter Name | Parameter Value | Parameter Name | Parameter Value |
|---|---|---|---|
| $k_d$ | 1.0 | $\tau_{fallback}$ | **3** |
| $k_t$ | 1.5 | $w_{fallback}$ | **1** |
| $\tau_1$ | 0.45 | $k_\sigma$ | **1.0** |
| $\tau_2$ | 0.65 | ***iterations*** | **0** |
| $\%added_d$ | 0.0 | ***#neighbors*** | **6** |
| $\%added_t$ | 1.0 | $w_d$ | **1.0** |
| ***acceptance%*** | 0.55 | $w_t$ | **0.0** |
| | | $w_r$ | **0.0** |

Table 3. Short-text and long-text data parameters optimization results for eight users.

#### 4.3.2. Second *Experiment (26 users)*

All the twenty six users entered Texts 1 and 2 for testing (see Appendix). The same six users were asked to try to log in using the other users' username/email and Text 1 (the same long-text used by all users in training) to provide imposter data (156 samples).

The optimization process for short-text and long-text data parameters for twenty six users resulted in the values shown in Table 4. For short-text data parameters, the values for $\%added_d$ and $\%added_t$ of 0.05 and 0.95 respectively mean that we relied by 95% on keys transitions features and by only 5% on keys durations features for short-text authentication for twenty six users. The value for *acceptance%* of 0.5 means that the system needs 50% matching between login data and enrolled data to consider the authentication successful.

For long-text data parameters, the value for *iterations* of 1 means that the outlier removal process in the long-text processing phase is required to be performed one time for this set of users. The values for $w_d$, $w_t$ and $w_r$ of 1.0, 0.0 and 0.0 respectively mean that we relied **only** on keys durations features for long-text authentication for twenty six users and keys transitions and ratios features were redundant, as the case with the first set of users.

| Short-text Parameters | | Long-text Parameters | |
|---|---|---|---|
| Parameter Name | Parameter Value | Parameter Name | Parameter Value |
| $k_d$ | 1.5 | $\tau_{fallback}$ | 4 |
| $k_t$ | 1.0 | $w_{fallback}$ | 5 |
| $\tau_1$ | 0.7 | $k_\sigma$ | 2.0 |
| $\tau_2$ | 0.55 | ***iterations*** | 1 |
| $\%added_d$ | 0.05 | ***#neighbors*** | 4 |
| $\%added_t$ | 0.95 | $w_d$ | 1.0 |
| ***acceptance%*** | 0.5 | $w_t$ | 0.0 |
| | | $w_r$ | 0.0 |

Table 4. Short-text and long-text data parameters optimization results for twenty six users.

### *4.4. Performance Evaluation*

The most commonly used metrics to evaluate the performance of a keystroke biometric system or any other biometric system are False Rejection Rate (FRR) and False Acceptance Rate (FAR). FRR is the percentage of legitimate users who are labeled as imposters and denied access. FRR denotes the system's specificity. FAR is the percentage of imposters who are allowed access to the system. FAR denotes the system's sensitivity. Accuracy is the percentage of legitimate users who are allowed access. It equals (100 – FRR). The average matching percentages for legitimate and imposter data are the total matching percentages for all users over the total attempts for legitimate and imposter data respectively. Tables 5 and 6 show the performance of our keystroke biometric system for both short-text and long-text data for eight users and twenty six users respectively.

| | Short-text data | Long-text data | | |
|---|---|---|---|---|
| | | Using Text 1 | Using Text 2 | Total |
| Average matching % for legitimate data | 100% | 100% | 89.6% | 94.8% |
| Average matching % for imposter data | 25% | 2.4% | | |
| Accuracy | **100%** | 100% | 87.5% | **93.8%** |
| FRR | **0%** | 0% | 12.5% | **6.2%** |
| FAR | **12.5%** | **2.1%** | | |

Table 5. Performance of our keystroke biometric system for both short-text and long-text data for eight users.

| | Short-text data | Long-text data | | |
|---|---|---|---|---|
| | | Using Text 1 | Using Text 2 | Total |
| Average matching % for legitimate data | 84.4% | 87.5% | 45.2% | 66.4% |
| Average matching % for imposter data | 22.5% | 1.12% | | |
| Accuracy | **100%** | 96.2% | 61.6% | **78.9%** |
| FRR | **0%** | 3.8% | 38.5% | **21.1%** |
| FAR | **27.6%** | **1.3%** | | |

Table 6. Performance of our keystroke biometric system for both short-text and long-text data for twenty six users.

As shown in Table 5, for short-text authentication for the first set of users, all legitimate users were authenticated correctly and 12.5% imposter tries were considered successful. For long-text authentication, 2.1% imposter tries were considered successful. All legitimate users were authenticated correctly when using the same long-text data for training and testing, and 12.5% legitimate tries were denied access when using different long-text data for training and testing.

As shown in Table 6, for short-text authentication for the second set of users, all legitimate users were also authenticated correctly but a higher percentage of imposter tries of 27.6% were considered successful. For long-text authentication, only 1.3% imposter tries were considered successful and 21.1% of legitimate tries were denied access. When using the same long-text data for training and testing, only 3.8% of legitimate tries were denied access.

### *4.5. Results and Discussion*

It is difficult to give a meaningful comparison of our approach with that of other studies as there is no unified data set under which the approaches can be compared and also no previous study has incorporated both short-text and long-text authentications before. We tested our


system's performance for short-text and long-text authentications over two sets of users, eight users and twenty six users, to emulate how our system would perform for a system of small number and larger number of users.

Firstly, for short-text authentication, our system had an FRR of 0% for both sets of users which means that no user who knows his login data (username/password combination) will be denied access to the system. Moreover, FAR of 12.5% and 27.5% for the two sets of users mean that some imposter tries could be accepted. Since keystroke dynamics is not the only mechanism for authentication but is integrated with password typing, so now an imposter needs to know the user's password and try to simulate his typing pattern too to gain access into the system.

For long-text authentication, FAR of 2.08% and 1.28% for the two sets of users mean that very few imposter tries would be allowed to gain access to the system. Using the same long-text for training and testing has very low FRR (0% and 3.84% for the two sets of users) which enhances the system's performance. However, making the users type the same long-text each time can cause inconvenience and that's why we needed to test the system performance when using different long-text data for training and testing. The system then had higher FRR values especially for the second set of users. The reason for the high rejection rate for the large set of users is the inconsistent and erratic typing patterns of some of our users when they have to copy a long-text they are not familiar with. This could be because English is a second language for them and some of them are inexperienced in English typing. This is clearly shown in the typing speed column of Table 2 where 15 users took more than 5 minutes to type a 600 character paragraph.

Our system showed that short-text authentication and long-text authentication are complementary to build a strong authentication system. Short-text authentication has higher FAR while long-text authentication has higher FRR, so each authentication type compensate for the shortcomings of the other. For instance, let's picture this scenario: When a user logs in, he provides his login information. Since our short-text authentication system has low FRR and high FAR, the user will mostly be allowed access to the system even when sometimes it's an imposter impersonating the user. Then, when the user is accessing the secured system, his long-text typing will be monitored and checked. Since our long-text authentication system for different long-text data in testing and training has low FAR and high FRR, the imposters will be detected (even when sometimes it's the legitimate user) and asked to enter the predefined long-text data (the same long-text data users entered during training) before they can continue accessing the secured system. Since the long-text authentication system has low FRR and FAR, imposters will be denied access to the system while legitimate users will be authenticated and their access to the secured system will be resumed.

Moreover, combining short-text and long-text authentications overcomes the inverse proportional relationship between FAR and FRR that most previous studies, as stated in Section 2, failed to overcome when they implemented single authentication systems that optimize one of the performance metric at the expense of the other, while using simple algorithms that are easy to implement and not computationally intensive.

The parameters optimization phase proved very important to our system. By tweaking the parameter values, we achieved the best performance for our system. The parameter optimization process also helped deciding which features are useful for short-text and long-text authentications. The optimization process for the short-text authentication system showed that we can rely solely on keys transitions features for the eight users system and by 95% for the twenty six users system. The optimization process for the long-text authentication system showed that we can rely solely on keys durations features for both eight users and twenty six users system and keys transitions and ratios features were redundant.

Furthermore, our two experiments showed that increasing the number of users reduced the system performance. For short-text authentication, FAR increased by a large value while FRR didn't change. For long text authentication, FRR increased by a very small value when using the same long-text data for training and testing and by a large value when using different long-text data for training and testing while FAR didn't change more or less.

All of the users, except for one, are in their twenties with an average age of about 25 years. 15 users are males and 11 users are females. For their long-text authentication testing, 60% of the male users got all their legitimate testing attempts accepted while 40% of them got only one attempt accepted. 63.6% of the female users got all their legitimate testing attempts accepted while 27.3% of them got only one attempt accepted and 9.1% got all attempts rejected. It appears that gender doesn't influence the accuracy of our system and it is more dependent on the users' typing experience, which can be identified by the time the users take to type the long-text data; the less time a user takes, the more experienced they are regardless of their typing errors, age or gender. Typing experience indicates less erratic typing style and hence more consistent typing patterns that allow better performance for our system.

## 5. Conclusions and Future Work

Keystroke dynamics is an inexpensive, non-intrusive biometric system that shows promising means for authentication for security systems. In this paper, we implemented a keystroke dynamics authentication system that deals with both short-text and long-text inputs to verify users identities. We used a statistical method as our short-text algorithm and a k-nearest


neighbor classification method as our long-text algorithm.

Our system has high specificity when dealing with short-text input and high sensitivity when dealing with long-text input. This shows that short-text authentication and long-text authentication are complementary to build a strong authentication system that cannot be easily compromised while remaining convenient and not annoying for the users.

The parameter optimization process that we introduced helped us decide which keystroke features are essential for user identification and which are redundant, and this increases the performance and efficiency and reduces the overhead of the authentication system.

We concluded that typing speed is the biggest parameter that indicates users' typing experience, regardless of their age, gender, or typing errors, and that typing experience influences the performance of our system.

Further investigation is necessary in the following areas: Firstly, different environmental conditions for training and testing must be explored. Secondly, data is required to be collected over long times to decide if users' typing patterns change and whether our system can handle these changes. Thirdly, experiments that evaluate our system over naïve users (users that are unaware that their typing patterns are captured) and imposters that are trying to mimic users' typing pattern are required. Moreover, two different fields should be explored: the first one is implementing and evaluating a keystroke dynamics biometric system for smart phones touch screens keyboards. The other field that requires exploring is implementing and evaluating a keystroke dynamics biometric system for other languages than English.

## 6. Acknowledgements

We would like to express our gratitude to all those who participated in the experiments.

## 7. Appendix

### Text 1:

*This is an Aesop fable about the bat and the weasels.*
*A bat who fell upon the ground and was caught by a weasel pleaded to be spared his life. The weasel refused, saying that he was by nature the enemy of all birds. The bat assured him he was not a bird, but a mouse, and thus was set free.*
*Shortly afterwards, the again fell to the ground and was caught by another weasel, whom he likewise entreated not to eat him. The weasel said that he had a special hostility to mice. The bat assured him that he was not a mouse, but a bat, and thus a second time escaped.*
*The moral of the story: it is wise to turn circumstances to good account.*

### Text 2:

*This is an Aesop fable about the wolf and the crane.*
*A wolf who had a bone stuck in his throat hired a crane, for a large sum, to put his head into his mouth and draw out the bone. When the crane extracted the bone and demanded the promised payment, the wolf grinding his teeth, exclaimed: "Why, you have surely already had a sufficient recompense, in having been permitted to draw out your head in safety from the mouth and jaws of a wolf"*
*The moral of the story: in serving the wicked, expect no reward, and be thankful if you escape injury for your pains.*

## 8. References

[1] S. D. Abualgasim and I. Osman. An Application of the Keystroke Dynamics Biometric for Securing PINs and Passwords. World of Computer Science and Information Technology Journal (WCSIT), 1(9):398–404, 2011.

[2] A. A. Ahmed and I. Traore. Biometric Recognition based on Free-text Keystroke Dynamics. IEEE Transactions on Cybernetics, 44(4):458–472, 2014.

[3] G. L. Azevedo, G. D. Cavalcanti, and E. C. Filho. Hybrid Solution for the Feature Selection in Personal Identification Problems through Keystroke Dynamics. International Joint Conference on Neural Networks, pp. 1947–1952, 2007.

[4] K. S. Balagani, V. V. Phoha, A. Ray, and S. Phoha. On the Discriminability of Keystroke Feature Vectors used in Fixed Text Keystroke Authentication. Pattern Recognition Letters, 32(7):1070–1080, 2011.

[5] N. Bartlow and B. Cukic. Evaluating the Reliability of Credential Hardening through Keystroke Dynamics. IEEE 17th International Symposium on Software Reliability Engineering (ISSRE'06), pp. 117–126, 2006.

[6] F. Bergadano, D. Gunetti, and C. Picardi. User Authentication through Keystroke Dynamics. ACM Transactions on Information and System Security (TISSEC), 5(4): 367–397, 2002.

[7] Ceker and S. Upadhyaya. Enhanced Recognition of Keystroke Dynamics using Gaussian Mixture Models, 2015.

[8] G. Dunn and B. S. Everitt, An Introduction to Mathematical Taxonomy. Courier Corporation, 2004.

[9] R. Giot, M. El-Abed, and C. Rosenberger. Greyc Keystroke: A Benchmark for Keystroke Dynamics Biometric Systems. IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS'09), pp. 1–6, 2009.

[10] D. Gunetti, C. Picardi, and G. Ruffo. Dealing with Different Languages and Old Profiles in Keystroke Analysis of Free Text. In AI* IA 2005: Advances in Artificial Intelligence. Springer Berlin Heidelberg, pp. 347–358, 2005.

[11] J. Hu, D. Gingrich, and A. Sentosa. A K-Nearest Neighbor Approach for User Authentication through Biometric Keystroke Dynamics. IEEE International Conference on Communications (ICC'08), pp. 1556–1560, 2008.

[12] D. Jurafsky and J. H. Martin. Speech and Language Processing. Prentice Hall, Englewood Cliffs, New Jersey 7632, 2000.

[13] K. S. Killourhy and R. Maxion. Comparing Anomaly-detection Algorithms for Keystroke Dynamics. In IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09), pp. 125–134, 2009.

[14] J.-W. Lee, S.-S. Choi, and B.-R. Moon. An Evolutionary Keystroke Authentication based on Ellipsoidal Hypothesis Space. In Proceedings of the 9th ACM Annual Conference on Genetic and Evolutionary Computation, pp. 2090–2097, 2007.

[15] H.-R. Lv, Z.-L. Lin, W.-J. Yin, and J. Dong. Emotion Recognition based on Pressure Sensor Keyboards. IEEE International Conference on Multimedia and Expo, pp. 1089–1092, 2008.

[16] J. Montalvão, C. A. S. Almeida, and E. O. Freire. Equalization of Keystroke Timing histograms for Improved Identification Performance. In IEEE International Telecommunications Symposium, pp. 560–565, 2006.

[17] R. V. Ponkshe and V. Chole. Keystroke and Mouse Dynamics: A Review on Behavioral Biometrics. International Journal of Computer Science and Mobile Computing, 4(2):341–345, 2015.

[18] J. A. Robinson, V. M. Liang, J. A. M. Chambers, and C. L. MacKenzie. Computer User Verification using Login String Keystroke Dynamics. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 28(2):236–241, 1998.

[19] D. Shanmugapriya and G. Padmavathi. Virtual Key Force – A New Feature for Keystroke. International Journal of Engineering Science and Technology (IJEST), 3(10):7738–7743, 2011.

[20] Y. Sheng, V. V. Phoha, and S. M. Rovnyak. A Parallel Decision Tree-based Method for User Authentication based on Keystroke Patterns. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 35(4):826–833, 2005.

[21] M. Villani, M. Curtin, G. Ngo, J. Simone, H. S. Fort, C. C. Tappert, and S.-H. Cha. Keystroke biometric recognition studies on long-text input over the internet. CSIS, Pace University, 2006.

[22] S. Yong, W. K. Lai, and G. Goghill. Weightless Neural Networks for Typing Biometrics Authentication. In Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin Heidelberg, pp. 284–293, 2004.

## Biographies

**Mahmoud AbdelDayem** received his B.Sc. degree in Computer and Control Engineering from Faculty of Engineering, Tanta University in 2010. Currently, he is a research scholar and a Master's degree student at Computer and Control Engineering Department, Faculty of Engineering, Tanta University, Egypt, under the supervision of Dr. Hamed Hemeda and Dr. Amany Sarhan. His research interests include Pattern Recognition, Machine Learning, Biometrics and Keystroke Dynamics.

**Hamed Hemeda** received his B.Sc. and M.Sc. degrees in Computer and Control Engineering from Faculty of Engineering, Tanta University and Faculty of Engineering, Mansoura University in 1995 and 1999, respectively. He was awarded the Ph.D. degree in Computer Engineering from Bretagne-Sud University, France in 2009. He is working now as lecturer at Computer and Control Engineering Department, Tanta University, Egypt. His interests are in the areas of: Human Computer Interaction (HCI) and Computer Networking.

**Amany Sarhan** received her B.Sc. degree in Electronics Engineering and M.Sc. in Computer Engineering from the Faculty of Engineering, Mansoura University in 1990 and 1997, respectively. She was awarded the Ph.D. degree as a joint research between Tanta University, Egypt and University of Connecticut, USA. She is working now as a full professor and head of Computer and Control Engineering Department, Tanta University, Egypt. Her interests are in the area of: Distributed Systems, Software Restructuring, Object-oriented Databases, and Image and video processing, GPU and Distributed Computations.

www.icgst.com

# Enhancing Grid Local Outlier Factor Algorithm for Better Outlier Detection

Eslam Mahmoud[1], Ahmed M. Elmogy[1,2], Amany Sarhan[1]
1. *Computers and Control Eng. Dept., Tanta Univ., Egypt*
2. *On leave to Arab East Colleges, Riyadh, KSA*
imhassan@imamu.edu.sa, Amelmogy@arabeast.edu.sa, amany.sarhan@f-eng.tanta.edu.eg

## Abstract
Detecting outliers in a large data set is a major data mining task. The existing approaches in this field are categorized into two main categories which are distance-based and density-based outlier detection approaches. Although, Local Outlier Factor (LOF) is considered as the most popular density-based algorithm, it still has some problems related to the speed and accuracy. Enhancing LOF algorithm has been the focus of many researchers working in this field. Among the improved versions of LOF, GridLOF has been proven to have a very good performance. This paper presents an enhancement to GridLOF algorithm by replacing one of its steps by a less complex step which reduces the complexity to be only $O(N)$ instead of $O(N^2)$ in a novel way. The simulation results show that the proposed algorithm outperforms GridLOF algorithm in terms of speed and accuracy.

***Keywords:*** *Outlier, outlier detection, data mining, LOF, GridLOF.*

***Nomenclature***

| | |
|---|---|
| LOF | Local Outlier Factor |
| LOCI | Local Correlation Integral |
| KLOF | Kernel Density-Based Local Outlier Factor |
| PNSR | Peak signal-to-noise ratio |

## 1.INTRODUCTION
Big data is a term used to describe the exponential growth of data, both structured and unstructured. In this case the data is so large or complex that traditional data processing techniques are inadequate. Big Data is mostly known to have three characteristics [1]: volume, variety, and velocity. Many factors contribute to the increase in data volume. These factors include, transaction-based data stored through years, unstructured data from social media, and increasing amounts of sensor and machine-to machine being collected. Velocity means that data must be dealt in a timely manner. As technology is evolving very rapidly, dealing with torrents of data in the real time becomes a challenge for most organizations. Finally, as data comes in all types of formats, managing different varieties of data is very important to consider while dealing with big data.

In order to discover patterns, unknown correlations, and other useful information in large data sets, big data analytics are employed [2]. Data mining is the most popular technique for data analytics [3]. This can be used in many areas such as sales, marketing, finance, medicine, supply chain management, and manufacturing. The process of mining patterns from data needs some primary operations. These processes include Data preparation, cleaning, and cleansing [4]. These operations deal with detecting and removing errors and inconsistencies (outliers) from data in order to improve the quality of data. While integrating multiple data sources in big data repositories such warehouses, the need for data cleaning increases significantly. Important issues about data preparation are discussed in [5].

An outlier is defined in [6] as an observation that deviates so much from other observations. Also, in [7], outlier is defined as an observation which appears to be inconsistent with the other members of the same data set. In [8], the terms outlining observation and outlier are used synonymously and are defined as a data observation that appears to deviate markedly from other members in the data set it occurs.

Outlier detection is the process of finding data objects with behaviors that are very different from expectation (outliers) [9]. Outlier detection methods are categorized into three types [3, 9, 10]: statistical methods, proximity-based methods, and clustering-based methods. The proposed work in this paper is some kind of proximity-based method. There are two types of proximity-based outlier detection methods: distance-based and density-based methods. A distance-based outlier detection method consults the neighborhood of an object, which is defined by a given radius [10]. An object is then considered an outlier if its neighborhood does not have enough other points. A density-based outlier detection method [11] investigates the density of an object and that of its neighbors. Here, an object is identified as an outlier if its density is relatively much lower than that of its neighbors.

Local Outlier Factor (LOF) is considered as the most popular density-based outlier detection [7, 12]. The main idea of LOF as proposed in [13] is using the relative density of an object against its neighbors. This relative density is used to assign a degree of being an outlier. This degree is called local outlier factor (LOF). Although there are many research efforts in the literature on simplifying, and enhancing LOF algorithm [14, 15, 16], more enhancements need to be done to deal with big data. LOF', LOF'', Grid LOF [17], and FastLOF [18] are examples of these efforts.

This paper proposes a new outlier detection algorithm based on enhancement of LOF algorithm. Despite the effort done to enhance the LOF algorithm, it still suffers from the complexity problem which appears clearly when dealing with big data. To improve the performance of the LOF, the proposed algorithm focuses on simplifying the step of finding the nearest neighbors nodes which is the major bottleneck in this algorithm. To prove the effectiveness of the proposed algorithm, time is chosen as a performance metric to assess the efficiency against the current algorithms.

The remainder of this paper is structured as follows. Section 2 provides related work about the outlier detection problem. A background about LOF algorithm is introduced in section 3. Section 4 presents the details of the proposed algorithm. Section 5 introduces the simulation results and provides an analytical discussion on the performance of the proposed algorithm. Section 6 introduces the conclusions and future work.

## 2. RELATED WORK

The importance of outlier detection comes from the fact that the deduced data can be translated into actionable information that can be used in many applications. These applications include but not restricted to fraud detection for credit cards, control systems, medical research, communication at runtime software, image sharing [32], intelligent transportation system, wireless sensor networks, and even human skin detection.

An extensive research effort has been seen in the literature tackling outlier detection. In [19], an overview of the existing outlier detection techniques in database management and data mining fields is provided. The introduced techniques have been classified according to different dimensions. Generally, distance-based [10] and density-based [11] outlier detection methods are the most important methods for outlier detection. The Major difference between the two methods is the granularity level [9] as distance-based methods give a higher level of granularity.

Distance-based methods are based on the nearest neighbor distances [9]. A point is considered as an outlier point if it has k-nearest neighbor distances larger than normal points. The detailed granularity level of distance-based methods gives more capacities in outlier analysis but with more computational cost [9]. There are many approaches for outlier detection in this category, such as cell-based approach [9], nested loop approach [3], and reverse nearest neighbor approach [9].

In real-world data sets, the structure of data is more complex as outliers are considered by their local neighborhoods or by the global data distribution [3]. Getting outliers by global data distribution is called density-based outlier detection. Thus outlier is detected by considering the density around each point in the data set. There are many approaches for outlier detection in this category such as Local Correlation Integral (LOCI) [9], histogram-based approaches [9], kernel density estimation approaches [9], and Local Outlier Factor (LOF) [9, 13]. Besides being the most popular density-based outlier detection approach, LOF is one of the simplest approaches for outlier detection [19].

In [13], each object is assigned a degree of being outlier. This degree is called the local outlier factor (LOF) of an object. This degree depends on how isolated the object is with respect to the surrounding neighborhood. LOF algorithm has been used in many applications such as cloud computing [20], network outlier detection [21], data clustering [22], fault analysis and detection [23], and steel plates fault diagnosis [24].

Towards enhancing LOF algorithm, many efforts have been seen in the literature. Kernel Density-Based Local Outlier Factor (KLOF) is an outlier detection algorithm which is based on LOF [25]. In [26], a hierarchical framework using approximated LOF is used for efficient anomaly detection. Also, an enhanced approach for LOF is proposed to be used for data mining purposes in [27]. The complexity of finding the nearest neighbors in LOF algorithm is $O(N^2)$ where the complexity of the algorithm itself is $O(N)$. Thus, many researchers tried to skip the step of finding the nearest neighbors in the LOF algorithm. In this path, LOF', LOF'', and GridLOF have been proposed in [17]. GridLOF is the most efficient and adaptive algorithm in calculating LOF value for each data objects in the data set. GridLOF algorithm also increases accuracy as it avoids some false identification that may occur in LOF. FastLOF has been also proposed in [18] to speed up the LOF computation. This is done by randomly dividing the dataset into groups. For each group, LOF is calculated and the point with LOF value greater than the defined threshold is identified (the threshold is the initially selected between 1.0 and 2.0). This process is repeated to find better neighbors. Although, FastLOF algorithm [18] can get outliers in any dataset, it cannot get all neighbors of a point as the data sets are divided randomly into groups.


## 3. BACKGROUND

As the proposed work in this paper is based on enhancing the LOF algorithm, some details about LOF and its enhanced version GridLOF are provided in this section. In [13], LOF is presented with the steps shown in figure 1.

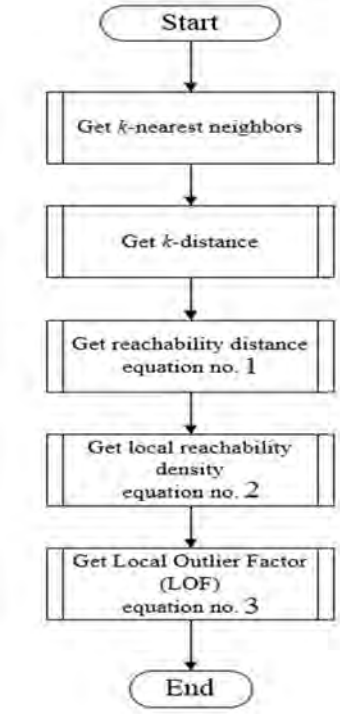



Figure 1: LOF Calculation Steps

1- Get *k*-nearest neighbors for an object *p*.
2- Get *k*-distance for an object *p*.
3- Get reachability distance of an object *p* with his *k*-nearest neighbors:

$$reach - dist_k(p,o) = max\big(k - distance(o), d(p,o)\big) \quad (1)$$

where *d(p, o)* is the distance between object *p* and its neighbor object *o*

4- Get local reachability density of an object p:

$$lrd_{MinPts}(p) = 1 \Big/ \left[\frac{\sum_{o \in N_{MinPts(p)}} reach - dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right] \quad (2)$$

This is based on the minimum points (*MinPts*) which is the nearest neighbors of *p*.

5- Get LOF for an object *p* as shown in figure 2 by:

$$LOF_{MinPts}(p) = \left[\frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}\right] \quad (3)$$

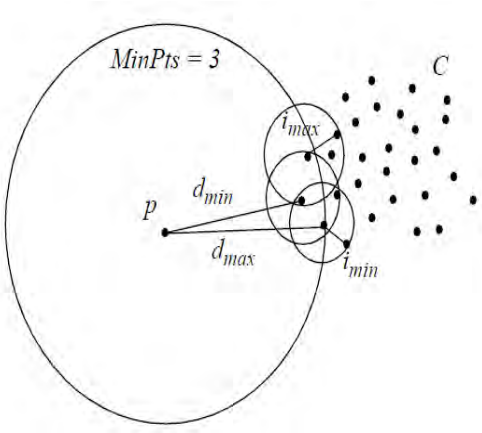



Figure 2: LOF for object *p*

In [17], GridLOF algorithm is proposed as an adaptive algorithm which prunes away the portion of dataset known to be non-outliers. First, each dimension of the data space is quantized into equal width intervals, resulting in a grid-based structure. Then, for each non empty grid cell c, the neighboring grid cells are examined and c is labeled as a boundary cell once a neighboring grid cell with less than or equal to the pre-defined threshold (σ) number of points residing in it is found. σ is a relatively a small number. In the extreme case, σ can be set to zero. Finally, only the LOF values of points inside boundary cells are calculated using above mentioned LOF (5 steps). Figure 3 illustrates the idea of GridLOF algorithm.

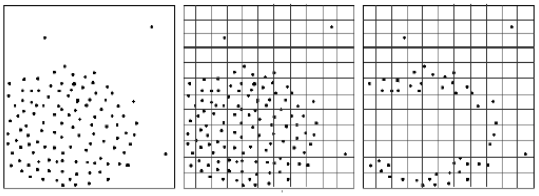


Figure 3: GridLOF algorithm

## 4. THE PROPOSED APPROACH

In GridLOF algorithm, the portion of dataset known to be non-outliers is prune away [17]. Local outlier factor (LOF) of the remaining points is then calculated. Although the overall cost for computing LOF can be reduced, GridLOF still has complexity of $O(N^2)$.

The proposed algorithm in this paper replaces the first step in the original LOF algorithm, which gets the *k*-nearest neighbors for an object *p*, by another method which gets the nearest neighbors in a more efficient and novel manner. The proposed work is based on re-defining each point $P_i$ by two values: $(w, \theta)$ (referred to an index point $P_0$ ) instead of the normal coordinates $(x, y)$where:

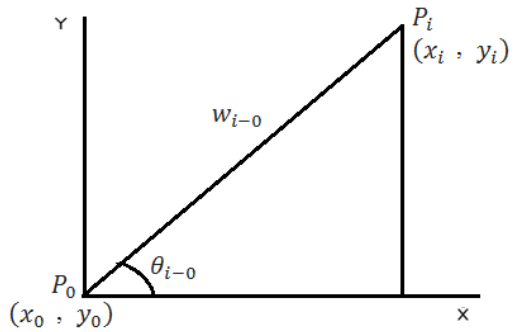


Figure 4: Re-defining each point by $(w, \theta)$

$w_{i-0}$: is the distance between $P_i$ and $P_0$ (as shown in figure 4) which is computed as:

$$w_{i-0} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad (4)$$

$\theta_{i-0}$: is the angle between line $P_0P_i$ and $X$ axis which is computed as:

$$\theta_{i-0} = \sin^{-1}\left(\frac{|y_i - y_0|}{w_{i-0}}\right) = \cos^{-1}\left(\frac{|x_i - x_0|}{w_{i-0}}\right) \quad (5)$$

For each point, a circle is drawn with radius $R$. The equations representing all the points within the circle are deduced which refer only to $W, \theta,$ and $R$. To get these equations, we need to define the following variables, as shown in figure 5:

$\varphi_{i-0}$: is the angle between line $P_0P_i$ and circle's tangent from $P_0$ which is computed as:

$$\varphi_{i-0} = \sin^{-1}\left(\frac{R}{w_{i-0}}\right) \quad (6)$$

$w_{i-0}^-$: is the minimum distance between $P_0$ and the circle which is computed as:

$$w_{i-0}^- = w_{i-0} - R \quad (7)$$

$w_{i-0}^+$ is the maximum distance between $P_0$ and the circle which is computed as:

$$w_{i-0}^+ = w_{i-0} + R \quad (8)$$

$\theta_{i-0}^-$: is the minimum angle between circle's tangent from $P_0$ and $X$ axis which is computed as:

$$\theta_{i-0}^- = \theta_{i-0} - \varphi_{i-0} \quad (9)$$

$\theta_{i-0}^+$: is the maximum angle between circle's tangent from $P_0$ and $X$ axis which is computed as:

$$\theta_{i-0}^+ = \theta_{i-0} + \varphi_{i-0} \quad (10)$$

Then, the hashed area in figure 5 can be described by the following g two equations:

$$w_{i-0}^- \le w_{n-0} \le w_{i-0}^+ \quad (11)$$

$$\theta_{i-0}^- \le \theta_{n-0} \le \theta_{i-0}^+ \quad (12)$$

Thus, a point $P_n$ lies in the hashed area if it satisfies equations 11&12.

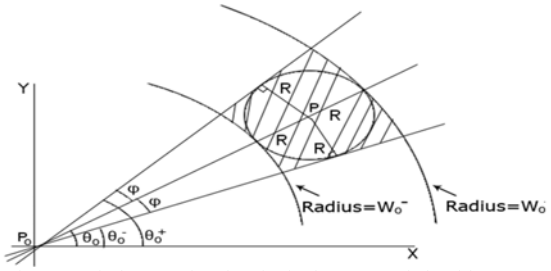


Figure 5: Circle around each point in the proposed algorithm

In order to define the equation of the circle of center $P$ shown in figure 5, we use four index points ($P_0$, $P_1$, $P_2$ and $P_3$), as shown in figure 6, instead of one point and thus:

$$w_{i-0}^- \le w_{n-0} \le w_{i-0}^+ \quad (13)$$

$$\theta_{i-0}^- \le \theta_{n-0} \le \theta_{i-0}^+ \quad (14)$$

$$w_{i-1}^- \le w_{n-1} \le w_{i-1}^+ \quad (15)$$

$$\theta_{i-1}^- \le \theta_{n-1} \le \theta_{i-1}^+ \quad (16)$$

$$w_{i-2}^- \le w_{n-2} \le w_{i-2}^+ \quad (17)$$

$$\theta_{i-2}^- \le \theta_{n-2} \le \theta_{i-2}^+ \quad (18)$$

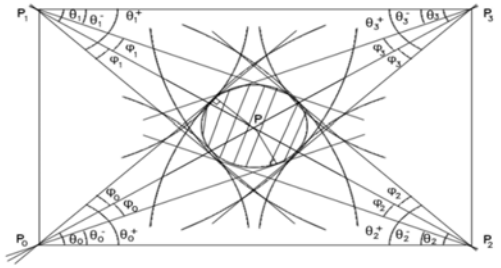


Figure 6: Defining a circle of center $P$ with four index points ($P_0$, $P_1$, $P_2$ and $P_3$)

$$w_{i-3}^- \le w_{n-3} \le w_{i-3}^+ \quad (19)$$

$$\theta_{i-3}^- \le \theta_{n-3} \le \theta_{i-3}^+ \quad (20)$$

A point $P_n$ lies in the hashed circle if it satisfies equations 13 through 20. To use the above idea in getting the k-nearest neighbor, for each point $P$ in the dataset, a circle with initial radius R is drawn. The number of points in this circle is deduced using the $(w, \theta)$ values which are calculated only once with each index point using equations 4 & 5. The radius of this circle is then increased until the number of points in the circle becomes equal to the required $k$-neighbors.

However, equations 13 through 20 don't represent the circle accurately when we work to get more number of neighbors (the equations represents the hashed shape in figure 5). For that, if we need to get the nearest ten neighbors for example, twelve neighbors are used to insure that we get the required ten neighbors inside the circle not inside the hashed shape. Thus two neighbors are added as an error ratio.

The complexity of the proposed algorithm, in comparison to the GridLOF algorithm, is only $O(N)$ while it is $O(N^2)$ in GridLOF algorithm [17]. In GridLOF, to find the nearest neighbors for a point, the distance between this point and the other points in the dataset is calculated. Thus, to get the nearest neighbors for whole dataset, $N * N$ calculations are performed (where $N$ is the size of the dataset). Whereas in the proposed algorithm, $(w, \theta)$ for each point is calculated only once with each index point and the radius $R$ is increased until getting needed neighbors. Thus, only $N$ calculations are done.

## 5. SIMULATIONS AND RESULTS

Our experiments are performed using Windows application written in C# running on Windows 7 with Intel® Core™ i7-3630QM @ 2.4 GHz processor and

16 GB RAM. Datasets are generated using *R* application [28] with sizes between 60K and 1000K and outlier of 0.1% to 0.5% of the dataset. Table 1 shows the different datasets. Implementation is made with grid size between 0.34*0.34 for 60K datasets, 0.5*0.5 for 125K datasets, 0.75*0.75 for 500K datasets, and 0.9*0.9 for 1000K datasets.

In these experiments, we will examine using the proposed step in the GridLOF algorithm. The main problem in the implementation is how to increase the radius of the circle. We started with initial value for *R* equals 2 and then it is increased by 2 (*R += 2*) for each iteration. Some points (such as outliers) in the dataset need many iterations to get nearest neighbors. Thus we need to limit the number of iterations (10 iterations is chosen) in order to save time and then we switch to the normal GRidLOF algorithm

Table 1 shows a comparison between GridLOF with and without the proposed modification using the execution time as an assessment metric. We have varied the size of the dataset and the number of outliers in it to test the performance at different environments. The last column in the table shows the improvement made by the proposed algorithm compared with the traditional GridLOF algorithm in terms of execution time for all test cases used.

Table 1: execution time results

| Dataset | | | Execution Time (Seconds) | | Improvement Percentage (difference) / (GridLOF) % |
|---|---|---|---|---|---|
| **Size** | **Outlier %** | **K (nearest neighbors)** | **GridLOF** | **Modified GridLOF** | |
| 60K | 0.1% | 10 | 240.628763 | 222.1957088 | 7.66 % |
| | | 15 | 248.961240 | 227.5250138 | 8.61 % |
| | | 21 | 252.651451 | 232.0782741 | 8.14 % |
| | 0.2% | 10 | 257.855749 | 239.8187169 | 7.00 % |
| | | 15 | 260.793917 | 245.9040646 | 5.71 % |
| | | 21 | 267.649309 | 249.8922932 | 6.63 % |
| | 0.5% | 10 | 240.403750 | 228.5240708 | 4.94 % |
| | | 15 | 244.709997 | 230.5451863 | 5.79 % |
| | | 21 | 249.880292 | 235.0324430 | 5.94 % |
| 125K | 0.1% | 10 | 510.830218 | 488.4689388 | 4.38 % |
| | | 15 | 516.922567 | 491.7331255 | 4.87 % |
| | | 21 | 524.575004 | 494.8103017 | 5.67 % |
| | 0.2% | 10 | 359.346554 | 339.7514326 | 5.45 % |
| | | 15 | 362.343725 | 341.9735596 | 5.62% |
| | | 21 | 370.563195 | 348.1349121 | 6.05 % |
| | 0.5% | 10 | 575.717929 | 554.9917438 | 3.60 % |
| | | 15 | 616.376255 | 562.2411583 | 8.78 % |
| | | 21 | 647.162016 | 566.5004021 | 12.46 % |
| 500K | 0.1% | 10 | 1754.697363 | 1631.817335 | 7.00% |
| | | 15 | 1815.866862 | 1635.92757 | 9.91% |
| | | 21 | 1818.568016 | 1641.982916 | 9.71% |
| | 0.2% | 10 | 1902.238802 | 1798.229853 | 5.47% |
| | | 15 | 1906.350037 | 1803.315144 | 5.40% |
| | | 21 | 1891.903211 | 1812.294657 | 4.21% |
| | 0.5% | 10 | 1350.35752 | 1275.496954 | 5.54 % |
| | | 15 | 1363.03296 | 1285.336517 | 5.70 % |
| | | 21 | 1378.05982 | 1292.721940 | 6.19 % |
| 1000K | 0.5% | 10 | 2886.046073 | 2759.082811 | 4.40% |
| | | 15 | 2976.336237 | 2783.228192 | 6.49% |
| | | 21 | 2985.340718 | 2810.691762 | 5.85% |

Figure 7 shows the execution time (in seconds) for both GridLOF and proposed modified GridLOF in case of data with size equal 60K and outlier percentage ranging from 0.1% to 0.5%. Results show that with increasing the number of nearest neighbors used in calculating LOF value for each point (K), the execution time increases. However, the execution time for the proposed modified GridLOF is still lower than that of normal GridLOF.

Figure 8 shows the execution time (in seconds) for both GridLOF and the proposed modified GridLOF in case of dataset with size equal 125K and outlier percentage ranging from 0.1% to 0.5%. Results show that the proposed modified GridLOF algorithm still outperforms the normal GridLOG algorithm.

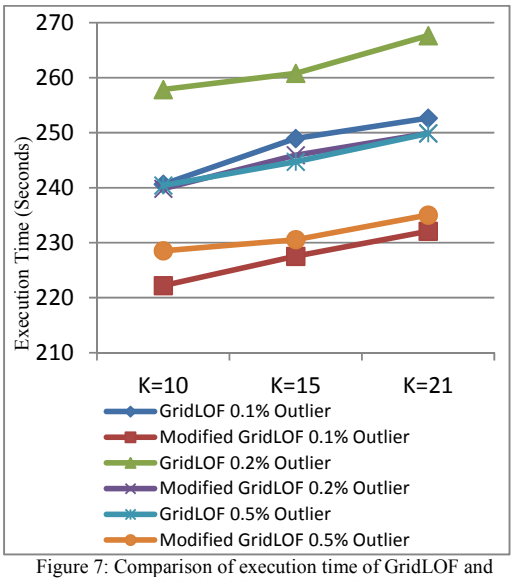




Figure 7: Comparison of execution time of GridLOF and Modified GridLOF at 60K dataset

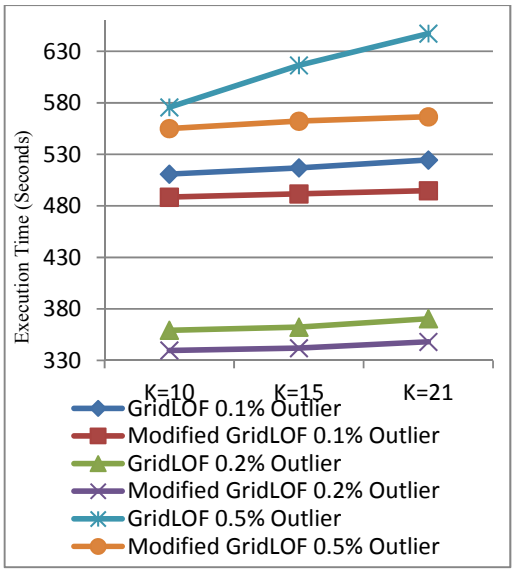



Figure 8: Comparison of execution time of GridLOF and Modified GridLOF at 125K Dataset

Figure 9 shows the superiority of the proposed GridLOF algorithm in case of dataset with size equal 500K and outlier percentage of from 0.1% to 0.5% compared with the normal GridLOF algorithm.

Figure 10 shows the superiority of the proposed GridLOF algorithm in case of dataset with size equal 1000K and outlier percentage 0.5% compared with the normal GridLOF algorithm.

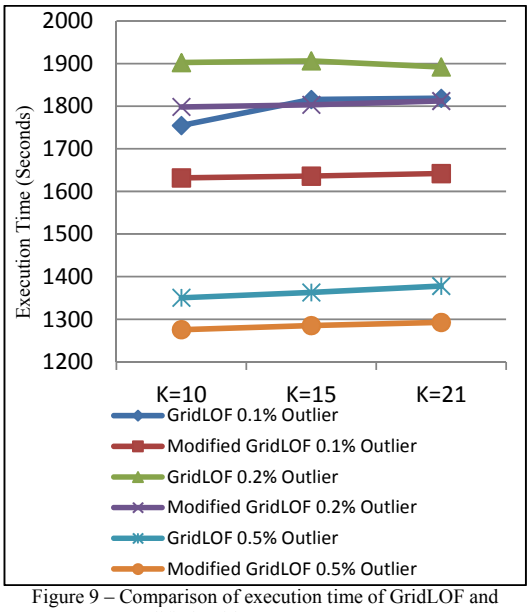



Figure 9 – Comparison of execution time of GridLOF and Modified GridLOF at 500K Dataset

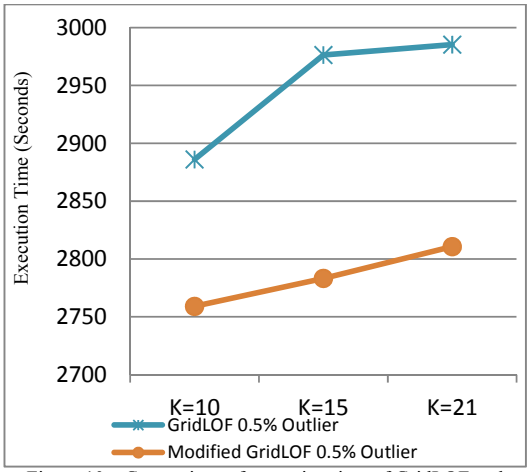



Figure 10 – Comparison of execution time of GridLOF and Modified GridLOF at 1000K Dataset

From Figure 7, 8, 9 & 10, experiments prove that the proposed modified GridLOF algorithm has a better performance than the normal GridLOF in all cases covered in the experiments and it is expected that this will continue in case of increasing data's size, Outlier percentage, or even K (k-nearest neighbors) value.

In order to check the performance of the proposed algorithm in terms of accuracy, new experiments are done. The proposed algorithm is used to detect and remove noise in images using median filter. The steps of applying the proposed algorithm on images are shown in figure 11.

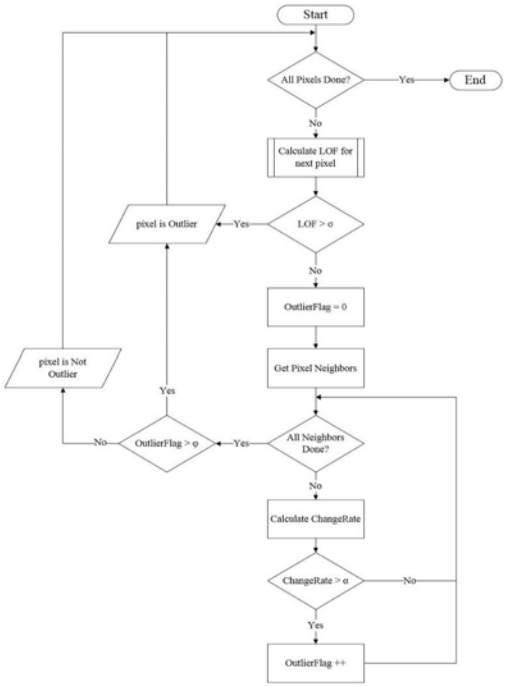



Figure 11: Steps to detect outlier in image

The value of $\varphi$ depends on needed level of saved useful information in the noisy image. In our experiments $\varphi = 0$ is used. A comparison of the proposed algorithm and the normal median filter [29] using PNSR (Peak signal-to-noise ratio). Figure 12 shows the original used image and the image with 5% noise.

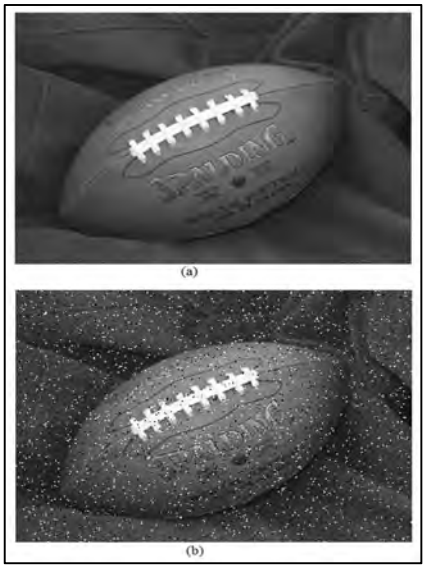



Figure 12: image 1 (a) original one (b) with 5% Salt & Pepper noise

Figure 13 shows the comparison results of filtering the noisy image (with 5% noise level) using the normal median filter and the proposed algorithm. The same experiments are done with 10%, and 20% noise levels and the PSNR is calculated in each case and the results are shown in figure 14.

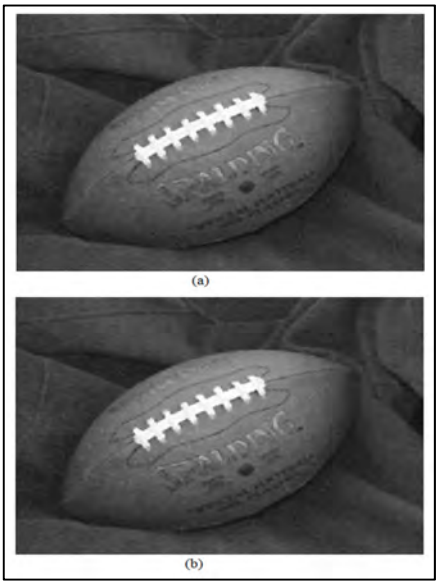



Figure 13: image 1 (5% noise) after apply median filter (a) Proposed algorithm (b) Normal median filter

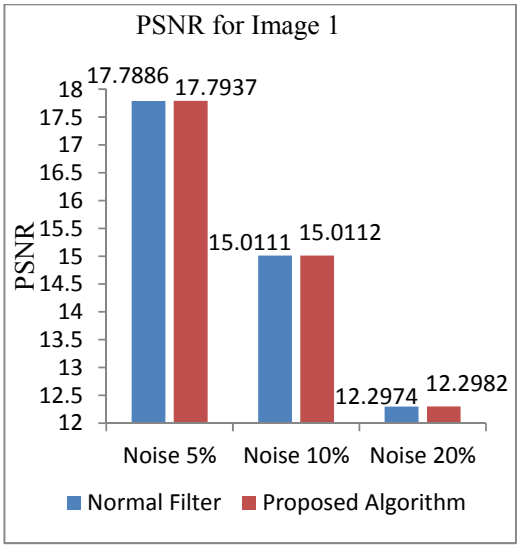



Figure 14: PSNR for image 1 with 5%, 10%, & 20% noise

Figure 14 shows that proposed algorithm has higher PSNR with 5%, 10%, 20% outlier percentages which proves that the proposed modified GridLOF algorithm has higher level of accuracy of detecting outliers in any dataset compared with normal median filter.

## 6. CONCLUSIONS

Outlier detection can be seen in many applications such as fraud detection for credit cards, control systems, medical research, image sharing, wireless sensor networks, and even human skin detection. This paper proposed a new outlier detection algorithm based on enhancement of LOF algorithm. The proposed algorithm focused on simplifying the step of finding nearest neighbors. Time and accuracy are chosen as performance metric to assess the

efficiency of the proposed algorithm. The proposed algorithm outperforms all kinds of LOF algorithm in terms of speed. The proposed algorithm is also used for image correction by detecting and removing outliers in the image.

Future work will focus on solving some issues that occurred during working with GridLOF algorithm such as increasing the radius of the circle which includes any point and iterations limits before getting the nearest neighbors.

## REFERENCES


[1] C. Eaton, D. Deroos, T. Deutsch, G. Lapis, P. Zikopoulos, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data IBM 2011.

[2] M. Barlow, Real-Time Big Data Analytics 2013.

[3] J. Han, M. Kamber, Data Mining: Concepts and Techniques 2000.

[4] E. Rahm, H. H. Do, Data Cleaning: Problems and Current Approaches, IEEE 2000.

[5] R. Cooley, B. Mobasher, J. Srivastava, Data Preparation for Mining World Wide Web Browsing Patterns, Knowledge and Information Systems, 1, pp, 5–32 1999.

[6] D. M. Hawkins, Identification of Outliers, Chapman and Hall 1980.

[7] R. A. Johnson, Applied Multivariate Statistical Analysis. Prentice Hall 1992.

[8] V. Barnett, T. Lewis, Outliers in Statistical Data. John Wiley 1994.

[9] C. C. Aggarwal, An Introduction to Outlier Analysis 2013.

[10] O. Z. Maimon, L. Rokach, Data Mining and Knowlege Discovery Handbook, Springer 2010.

[11] L. Duan, L. Xu, Y. Liu, J. Lee, Cluster-based outlier detection, Annals of Operations Research (Volume 168, Issue 1 , pp 151-168) Springer 2008.

[12] W. Jin, A. K. H. Tung, J. Han, Mining top-n local outliers in large databases, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (Pages 293-298) ACM 2001.

[13] M. M. Breunig, H. Kriegel, R. T. Ng, J. Sander, LOF: Identifying Density-Based Local Outliers, Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX 2000.

[14] J. Tang, Z. Chen, A. W Fu, D. W. Cheung, Enhancing Effectiveness of Outlier Detections for Low Density Patterns, Advances in Knowledge Discovery and Data Mining (Volume 2336 of the series Lecture Notes in Computer Science pp 535-548) Springer Berlin Heidelberg 2002.

[15] S. Jiang, Q. Li, K. Li, H. Wang, Z. Meng, GLOF: a new approach for mining local outlier, Machine Learning and Cybernetics, International Conference on (Volume:1 ), IEEE 2003.

[16] Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, Pattern Recognition Letters (Volume 24, Issues 9–10) Elsevier Science 2003.

[17] A. L. Chiu, A. W. Fu, Enhancements on Local Outlier Detection, Database Engineering and Applications Symposium, 2003. Proceedings. Seventh International, IEEE 2003.

[18] M. Goldstein, FastLOF: An Expectation-Maximization based Local Outlier Detection Algorithm, 21st International Conference on Pattern Recognition, IEEE 2012.

[19] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys 2009.

[20] T. Huang, Y. Zhu, Q. Zhang, Y. Zhu, D. Wang, M. Qiu, L. Liu, An LOF-based Adaptive Anomaly Detection Scheme for Cloud Computing, IEEE 37th Annual Computer Software and Applications Conference Workshops 2013.

[21] E. Schubert, A. Zimek, H. Kriegel, Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection, Springer 2012.

[22] V. SALTENIS, Data Clustering Based on Maximization of Outlier Factor, Journal of Global Optimization, Springer 2006.

[23] Y. Zhao, F. Balboni, T. Arnaud, J. Mosesian, R. Ball, B. Lehman, Fault Experiments in a Commercial-Scale PV Laboratory and Fault Detection Using Local Outlier Factor, Photovoltaic Specialist Conference (PVSC) IEEE 40th 2014.

[24] Z. Zhao, J. Yang, W. Lu, X. Wang, Application of Local Outlier Factor method and Back-Propagation Neural Network for steel plates fault diagnosis, Control and Decision Conference (CCDC) 27th IEEE 2015.

[25] M. Dandan, Q. Xiaowei, W. Weidong, Anomalous Cell Detection with Kernel Density-Based Local Outlier Factor, China Communications (Volume:12 Issue: 9), IEEE 2015.

[26] L. Xu, Y. Yeh, Y. Lee, J. Li, A Hierarchical Framework Using Approximated Local Outlier Factor for Efficient Anomaly Detection, The 3rd International Workshop on Sensor Networks for Intelligence Gathering and Monitoring (SNIGM), Procedia Computer Science 2013.

[27] V. Bhatt, K. G. Sharma, A. Ram, An Enhanced Approach for LOF in Data Mining, Proceedings of 2013 International Conference on Green High Performance Computing, IEEE 2013.

[28] R, data analysis software, more info. http://www.inside-r.org/

[29] I. Pitas, A. N. Venetsanopoulos, Order statistics in digital image processing, Proc. IEEE, vol. 80, no. 12, Decamber 1992.

## Tables



## Figures



## Biographies

Eslam Mahmoud, received the B.Sc degree in Computer Science and Automatic Control from the Faculty of Engineering, Tanta University, in 2009. He is working now as Senior Software Developer. His research interests include; Software Engineering, Database and Big Data.

Ahmed M. Elmogy received his B.Sc., and M.Sc from Computers & Control Dept., Faculty of Eng., Tanta Univ., Egypt in 1998, and 2003. He is awarded his Ph.D. from ECE Dept., Waterloo Univ., Canada. He is currently working as an Assistant Professor, Computers & Control Dept., Faculty of Eng., Tanta Univ., Egypt. His research interests include; Data Mining, Artificial Intelligence, Cryptography, and Robotics.

Amany Sarhan, received the B.Sc degree in Electronics Engineering, and M.Sc. in Computer Engineering from the Faculty of Engineering, Mansoura University, in 1990, and 1997, respectively. She awarded the Ph.D. degree as a joint research between Tanta Univ., Egypt and Univ. of Connecticut, USA. She is working now as a Full Prof. and head Computers and Control Dept., Tanta Univ., Egypt. Her research interests include; Distributed Systems, Software Restructuring, Object-oriented Databases, and Image and video processing, GPU and Distributed Computations.



www.icgst.com

# Minimum Average Scheduling Algorithm, MASA, Performance Boosting Approach

Kamal ElDahshan, Afaf Abd El-kader, Nermeen Ghazy
*Dept. of mathematics, Computer science Division, Faculty of science, Al-Azhar University*
*Cairo, Egypt*
[dahshan]@azhar.edu.eg, [afaa2islam, nermeen_ghazy89]@yahoo.com
http://www.azhar.edu.eg

## Abstract
Scheduling is the process of allocating tasks to resources in order to optimize some objective function. There have been many algorithms used to schedule tasks on their resources .One of these scheduling algorithms is the Enhanced Max-min task scheduling algorithm which selects a task with average execution time(average or nearest greater than average) and assigns it to a corresponding resource producing minimum completion time. A drawback of this algorithm is that when some average is too large, it causes the makespan to increase. This paper proposes an efficient scheduling algorithm MASA (Minimum Average Scheduling Algorithm) which selects $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7) tasks with minimum averages execution time (minimum averages or nearest greater than minimum averages) and then assigns these task(s) to the corresponding resources producing a minimum completion time. Experimental results ameliorate the makespan.

***Keywords:*** *Scheduling, Max – min algorithm, Improved Max-Min Algorithm, Enhanced Max-min Task Scheduling Algorithm, Distributed system, Scheduling algorithm, Minimum Average Scheduling Algorithm.*

***Nomenclature***

| | |
|---|---|
| M | number of resources |
| N | number of tasks |
| CPU | central processing unit |
| MASA | minimum average scheduling algorithm |
| T | task |
| R | resource |
| $C_{ij}$ | completion time for task $T_i$ on resource $R_j$ |
| $E_{ij}$ | execution time for task $T_i$ on resource $R_j$ |

## 1. Introduction
Grid computing systems enable sharing large-scale resources among millions of computer systems such as the Internet. The grid infrastructure involves four levels. First: the foundation level, it includes the physical components. Second: the middleware level, it is actually the software responsible for resource management, task execution, task scheduling, and security. Third: the service level, it provides users with efficient services, Fourth: the application level, it contains the services such as operational utilities and business tools.
Scheduling has become one of the major research objectives, because it directly increases the performance of grid applications [1]. It manages jobs to allocate appropriate resources by using scheduling algorithms and policies [2].
Scheduling can be classified as either static or dynamic. In static scheduling, the information regarding all the resources as well as the tasks is assumed to be known in advance. Furthermore, each task is assigned once to a resource. On the other hand, in dynamic scheduling, the task allocation is done while the application executes where it is not possible to find execution times. Several heuristic algorithms for grid task scheduling have been developed to improve grid performance [3], [4].
The aim of the scheduling problem is to optimize some objective functions such as makespan, CPU utilization, throughput, turnaround, response time, waiting time or fairness [5].

- Makespan: the maximum completion time
- CPU utilization: keeping the CPU as busy as possible
- Throughput: the number of processes that are completed per unit of time
- Turnaround: the time between starting and completion
- Response time: the time from submission of a task to the first response

- Waiting time: the amount of time that is waiting in the ready queue
- Fairness: giving each process a fair share of the CPU

The main contribution of this work is to introduce an efficient algorithm for scheduling tasks on resources with minimizing the makespan.

The remainder of the paper is organized as follows: Section (2) focuses on some related work Section (3) focuses on Enhanced Max-min Task Scheduling Algorithm Section (4) presents the proposed algorithm MASA Section (5) describes Experimental data Section (6) explains Results analysis Section (7) concludes the paper and presents future work.

## 2. Related work

Many algorithms have been used to schedule tasks on their resources:

**FCFS (First Come, First Served) algorithm:** it is a non- preemptive algorithm. jobs are come on first executes and served first, it is so easy because it selects first process whatever has long or short execution time so the average waiting time is high which is poor in performance.

**SJF (Shortest Job First) algorithm:** Shortest Job First algorithm is a non – preemptive algorithm where in the ready queue, the short jobs are executed first. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next execution time. If there is two processes have the same execution time, it is used FCFS scheduling.

**SRTF (Shortest remaining time first) algorithm:** it is a preemptive SJF algorithm; it preempts the currently executing process while a non-preemptive SJF algorithm will allow to finish the currently running process first.

**Round Robin algorithm:** Round Robin is the preemptive process scheduling algorithm. It has a quantum which is a fixed time provided for each process. While it is executed for given time period it is preempted and other process executes for given time period. It is used context switching to save states of preempted processes.

**Priority scheduling algorithm:** it is a non-preemptive algorithm. Each process is assigned a priority. The priority can be decided based on memory requirements, time requirements or any other resource requirement. The Process with highest priority will be executed first and so on. If the processes have the same priority, it will be executed on first come first served basis.

**Multiple-Level Queues Scheduling:** Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics. Each queue can have its own scheduling algorithm and priorities.

**Min–Min algorithm:** This algorithm finds the task which has a minimum execution time and assigns the task to the resource that produces minimum completion time. The ready time of the resource is updated. This procedure is repeatedly executed until all unmapped tasks are scheduled [6], [7], [8], [9].

**Max–Min algorithm:** The Max-Min scheduling algorithm schedules the larger task first. The ready time of the resource is then updated. This procedure is repeated until all-unscheduled tasks are assigned. If there is only one long task, the Max-min algorithm executes many short tasks concurrently with the long task and the makespan of the system is most likely determined by the execution time of the long task so Max-Min scheduling algorithm gives better results than Min–Min algorithm [6], [7], [8], [9].

**RASA algorithm:** this algorithm uses the Max-Min, Min-Min algorithms. If the number of available resources is odd, the Min-min algorithm is applied to assign the first task, whereas the number of available resources is even the Max-min algorithm is applied.
The remaining tasks are assigned to their appropriate resources by one of the two algorithms alternatively. If the Min-min algorithm is applied to assign the first task to resource, the next task will be assigned by the Max-min algorithm. In the next round the task assignment begins with an algorithm different from the last round. So if the first round begins with the Max-min algorithm, the second round will begin with the Min-min algorithm [8].

**Improved Max-Min Algorithm:** This algorithm uses the advantages of Max-Min and solves its disadvantages. It is concerned with the number of the resources and the tasks. It is based on the expected execution time instead of completion time. This algorithm calculates the expected completion time of the tasks on each resource. Then the task with the maximum expected execution time (Largest Task) is assigned to a resource that has the minimum overall completion time (Slowest Resource). Then, this scheduled task is removed from meta-tasks and all corresponding times are updated and then the traditional max-min algorithm is applied to the remaining tasks. The allocation of the slowest resource to longest task allows availability of high-speed resources for finishing other small tasks, which achieves the shortest makespan of submitted tasks on available resources [10].

**Enhanced Max-min Task Scheduling Algorithm:** This algorithm introduced a unique modification of Improved Max-min task scheduling algorithm. It is based on the


expected execution time instead of completion time. It assigns the task with average execution time (average or nearest greater than average Task) to the slowest resource produces minimum completion time. This reduces overall makespan and balance load across resources [11].

This algorithm is provably described below.

## 3. Enhanced Max-min Task Scheduling Algorithm

Enhanced Max-min algorithm is based on the expected execution time instead of completion time. In the set of tasks to be scheduled, the largest task may be too large compared to other tasks which cause increasing makespan. This occurs because first; the largest task is executed by the slowest resource that has the minimum overall completion time while other tasks are executed by faster resources. Hence, the Enhanced max-min algorithm selects a task with (average or nearest greater than average) execution time and assign it to the slowest resource produces minimum completion time. Then, this task is removed from meta-tasks and the all calculated times are updated. The traditional max-min algorithm is then applied to the remaining tasks [11].

## 4. The proposed algorithm: MASA (Minimum Average Scheduling Algorithm)

Sometimes in the matrix of execution time the average of some resource may be very large compared to other averages, applying the Enhanced Max-min algorithm will produce a great makespan. The proposed algorithm improves the Enhanced Max-min algorithm, instead of selecting average or nearest greater than an average task. MASA selects the $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7)tasks that have (minimum average execution time or nearest greater than minimum average execution time). These tasks are then assigned to the corresponding resources and the traditional Max-min algorithm is applied. MASA decreases makespan because it is does not depend on the largest average and assigns minimum average tasks to faster resources which produce minimum completion time.

### Minimum Average Scheduling Algorithm, MASA

The MASA algorithm is as follows

*1: For all tasks $t_i$ in Meta task*
*2: For all resources $R_j$*
*3: $C_{ij} = E_{ij} + r_j$*
*4: Compute average execution time for all resources*
*5: Select k= $\lfloor m/7 \rfloor$ task(s) with minimum averages*
*6: Assign selected task(s) to the corresponding resources $R_j$ that gives minimum completion time*
*7: Delete selected task(s) from Meta task*
*8: Update completion times for corresponding resources*
*9: While there are tasks in Meta task*
*10: For each task find earliest completion time and the resource that obtains it*
*11: Find the task $T_k$ with maximum earliest completion time*
*12: Assign task $T_k$ to the corresponding resource $R_j$ that gives minimum completion time*
*13: Update $C_{ij}$ for all i*
*14: End while*

The complexity of the proposed algorithm is $O(mn^2)$ as that of the Enhanced Max-min algorithm, where m is the number of resources in the system and n is the number of tasks which should be scheduled to be executed.

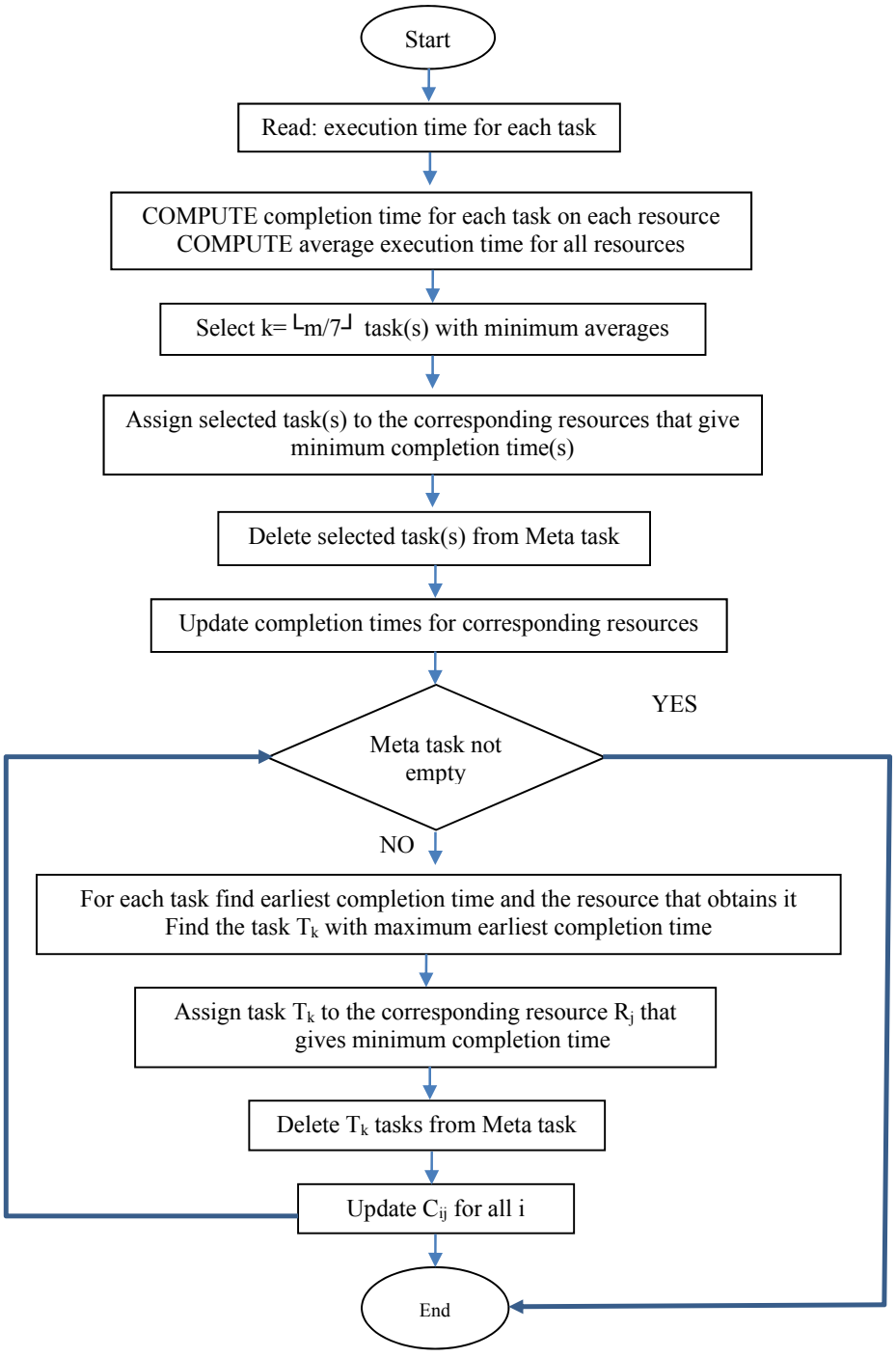
### A Comparison between the Enhanced Max-min Algorithm and Minimum Average Scheduling Algorithm

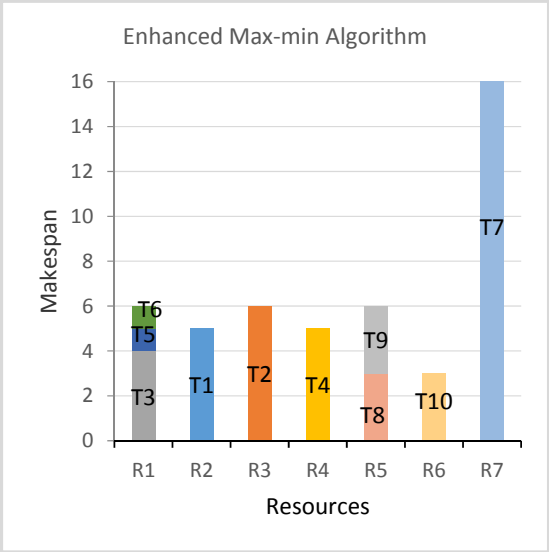
Assume that task scheduler has meta-tasks and resources with execution times for each task on each resource given below in Table 1.

Table1. Execution times of tasks

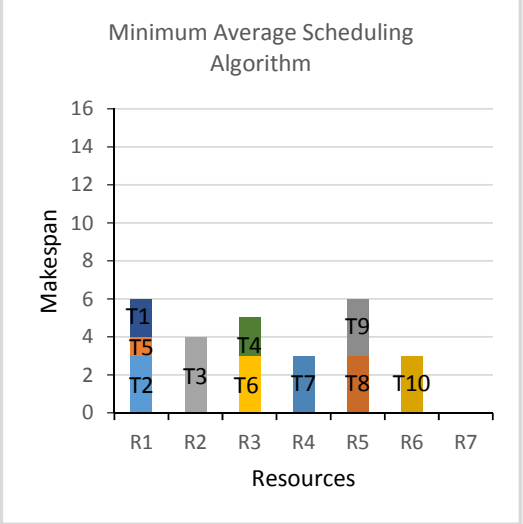
| R / T | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|
| **T1** | 2 | 5 | 5 | 6 | 2 | 8 | 13 |
| **T2** | 3 | 2 | 6 | 7 | 9 | 9 | 17 |
| **T3** | 4 | 4 | 4 | 8 | 8 | 10 | 12 |
| **T4** | 5 | 6 | 2 | 5 | 7 | 9 | 18 |
| **T5** | 1 | 2 | 1 | 4 | 6 | 7 | 19 |
| **T6** | 1 | 1 | 3 | 8 | 10 | 6 | 12 |
| **T7** | 2 | 3 | 2 | 3 | 10 | 8 | 16 |
| **T8** | 5 | 7 | 4 | 9 | 3 | 10 | 13 |
| **T9** | 4 | 9 | 8 | 10 | 3 | 10 | 20 |
| **T10** | 3 | 11 | 5 | 10 | 2 | 3 | 20 |

Minimum Average Scheduling algorithm, MASA, Flowchart






(a)
Figure 1: (a) Gantt chart of Enhanced Max –min algorithm




(b)
Figure 1: (b) Gantt Chart of Minimum Average Scheduling Algorithm

Figure 1 describes the makespan using both the Enhanced Max–min algorithm and the MASA algorithm.
The Enhanced Max–min algorithm produces a makespan=16 ms while MASA produces a makespan=6 ms.
The results show that MASA produces a makespan smaller than that of the Enhanced Max-min algorithm.

## 5. Experimental Data

A simulation is made for the MASA algorithm and the Enhanced Max-min algorithm to analyze the performance of each algorithm. The simulation is written using the C++ language. The model consists of m resources, n tasks with m varying from 50 to 400 and n varying from 300 to 2700. The values of execution times of tasks are chosen randomly.
To present the advantages of the MASA algorithm against the Enhanced Max-min algorithm; Figure 2 plots the makespan versus the number of tasks for both the Enhanced Max-min algorithm and the MASA algorithm. Figure 3 plots the makespan versus the number of resources for both the Enhanced Max-min algorithm and the MASA algorithm.

In Figure 2, the number of tasks is varying from 300 to 2700 and the number of resources is constant and equals to 100. By taking 1500 tasks, the Enhanced Max-min algorithm gives makespan= 445 ms while MASA algorithm gives makespan= 294 ms. By taking 2700 tasks the makespan for Enhanced Max-min algorithm is 1106 ms while the makespan for MASA algorithm is 781 ms.
In Figure 3, the number of tasks is constant and equals to 1000 while the number of resources varies from 50 to 400. It is observed the makespan results of MASA algorithm is varying between 150 ms to 49 ms whereas the makespan results of the Enhanced Max-min algorithm is varying from 205 ms to 195 ms.

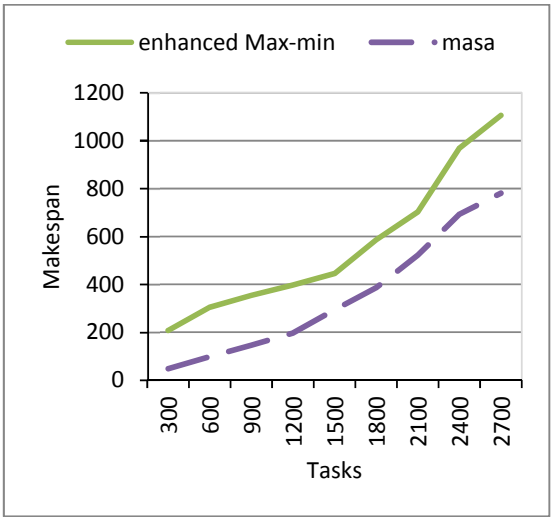



Figure 2:The makespan versus the number of tasks for both Enhanced Max-min algorithm and Minimum Scheduling Algorithm

## 6. Results analysis

MASA algorithm selects minimum averages tasks and assigns them to the corresponding resources. The number of selected tasks depends on the number or resources. Testing different fractions m/2, m/3… m/10 showed that the fraction m/7 gives the best result.
Suppose that the number of tasks=1000, the number of resources=100, the Enhanced Max-min algorithm produces makespan=204 ms, in MASA by applying different fractions the results are showing in table 2.

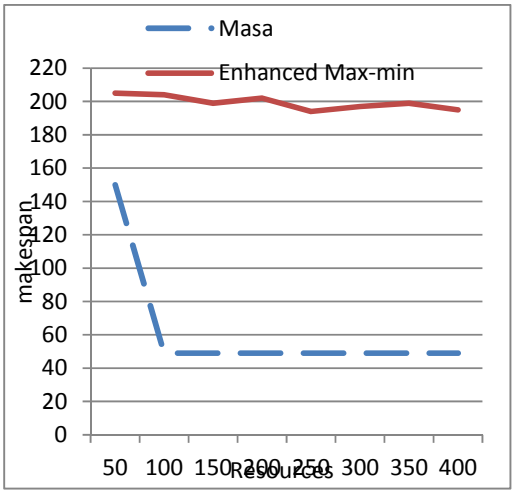



Figure 3: The makespan versus the number of resources for both Enhanced Max-min algorithm and MASA algorithm

Table2. The makespan of different fractions

| fraction | makespan |
|---|---|
| **m/2** | 71 |
| **m/3** | 61 |
| **m/4** | 59 |
| **m/5** | 52 |
| **m/6** | 50 |
| **m/7** | 49 |
| **m/8** | 49 |
| **m/9** | 49 |
| **m/10** | 49 |

By choosing m/2 tasks, MASA produces a makespan= 71 ms, m/3 tasks produces a makespan= 61 ms, m/4 tasks produces a makespan= 59 ms, m/5, m/6, m/7 tasks produces 52, 50, 49 ms.
It is noted that the makespan decreases as the fraction decreases from m/2 to m/7. Makespan is decreased when the number of selected tasks decreases, but at the range between m/7,...m/10 the makespan remains stable and results do not change almost.

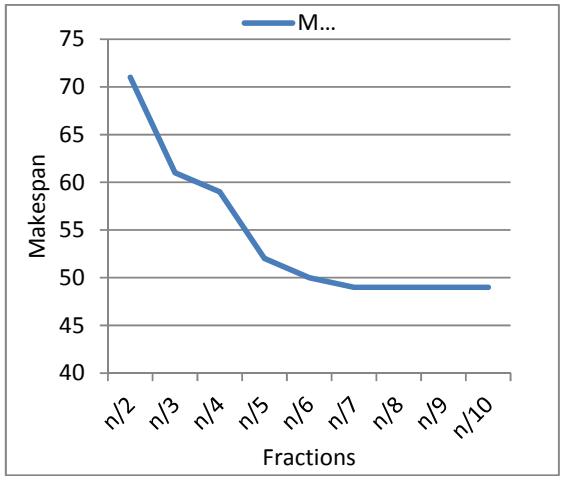



Figure4: The makespan of MASA for different fractions

It is noted from the figure that makespan decreases and then become stable. Hence, there is a breakpoint between the two fractions m/6, m/7. Since the number of tasks is an integer number, so ⌐m/7¬ is selected.

## 7. Conclusions

The Enhanced Max-min algorithm assigns the task with average execution time (average or nearest greater than average) to the slowest resource produces minimum completion time. When all averages are small compared to some average, the makespan increases. This paper proposed the MASA algorithm which selects ⌊m/7⌋ tasks with minimum averages execution times or nearest greater than average and assigns them to the corresponding resources producing minimum completion time. Comparing the makespan produced by the two algorithms; the results show that the MASA algorithm improves the makespan.

## Future Work

The proposed algorithm selects ⌊m/7⌋ tasks with minimum averages execution times or nearest greater than average and assigns them to the corresponding resources producing minimum completion time, it may be applied in parallel computing, cloud computing, distributed systems, operating systems.

## 8. References

[1] Naglaa M. Reda, A. Tawfik, Mohamed A.Marzok, Soheir M. Khamis, "Sort-Mid tasks scheduling algorithm in grid computing", Journal of Advanced Research, Vol. 6(6), pp. .987–993, 2015.

[2] A. Chandak, B. Sahoo, A. Turuk, "An overview of task scheduling and performance metrics in grid computing", International Journal of Research and Reviews in Computer Science, Vol. 2(2), pp. 30–33 2011.

[3] Braun TD, Siegel HJ, Beck N, Boloni LL, Maheswaran M, Reuther AL, et al, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed

computing systems", J Parallel Distrib Compute, Vol. 61(6), pp.810–37, 2011.
[4] Elzeki OM, Rashad MZ, Elsoud MA, "Overview of scheduling tasks in distributed Computing systems ", Int J Soft Comput Eng, Vol. 2(3), pp.470–475, 2012.
[5] Neetu Goel, R.B. Garg, "A Comparative Study of CPU Scheduling Algorithms", International Journal of Graphics & Image Processing, Vol. 2(4), pp. 245-251, 2012.
[6] El-Sayed T. El-kenawy, Ali Ibraheem El Desoky, Mohamed F. Al-rahamawy, "Extended Max-Min Scheduling Using Petri Net and Load Balancing", International Journal of Soft Computing and Engineering (IJSCE), Vol. 2(4), pp. 198-203, 2012.
[7] Pinal Salot, "A Survey of various scheduling algorithm in cloud computing environment", IJRET - International Journal of Research in Engineering and Technology, Vol. 2(2), pp.131-135, 2013.
[8] Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, Vol. 3(4), pp. 91-99, 2009.
[9] D. Maruthanayagam, Dr. R. Uma Rani, "Enhanced Ant Colony System Based on RASA Algorithm in Grid Scheduling", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2(4), pp.1659-1674, 2011.
[10] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, Vol. 50(12), pp.22-27, 2012.
[11] Upendra Bhoi, Purvi N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering & Management, Vol. 2(4), pp. 259-264, 2013.

## Biographies

**Prof. Kamal Abdelraouf ElDahshan** He is a professor of Computer Science and Information Systems at Al - Azhar University in Cairo, Egypt. An Egyptian national and graduate of Cairo University, he obtained his doctoral degree from the Universitè de Technologie de Compiègne in France, where he also taught for several years. During his extended stay in France, he also worked at the prestigious Institute National de Tèlècommunications in Paris. Professor ElDahshanâAZs extensive international research, teaching, and consulting experiences have spanned four continents and include academic institutions as well as government and private organizations. He taught at Virginia Tech as a visiting professor; he was a Consultant to the Egyptian Cabinet Information and Decision Support Centre (IDSC); and he was a senior advisor to the Ministry of Education and Deputy Director of the National Technology Development Centre. Professor ElDahshan is a professional Fellow on Open Educational Resources as recognized by the United States Department of State and an Expert at ALECSO as recognized by the League of Arab States.

**Dr. Afaf Abd El-Kader Abd EL-Hafiz** is currently lecturer at University of Al-Azhar. She is doing her research work in Scheduling algorithms.

**Nermeen Alaa Eldin Ghazy** She received her B.Sc. from Faculty of Science, AL-Azhar University at 2011. and Ms. Ghazy is preparing M.Sc in scheduling algorithms.



www.icgst.com

# Arabic Full Text Diacritization using Light Layered Approach

Aya S. M. Hussein, Mohsen A. A. Rashwan, Amir F. Atiya
*Faculty of Engineering,Cairo University, Egypt*
ayasalah_89@yahoo.com - ayasalah_89@cu.edu.eg, mrashwan@rdi-eg.com, amir@alumni.caltech.edu
http://www.eng.cu.edu.eg

## Abstract

Text diacritic restoration is a very vital problem for languages that use diacritics in their orthography systems. Actually, it plays an important role for improving the performance of many NLP tasks. In this paper, we handle the problem of Arabic text diacritization; such that our system diacritizes input sequence of words both morphologically and syntactically. The operation of the system is divided into three layers; each layer handles a specific problem. To evaluate the performance of the system, we used the benchmark LDC Arabic Treebank datasets used by the state of the art systems, for the sake of fair comparison. Besides, we also used an extra test set to give an indication of the real performance of the system on any totally independent data set.
For morphological diacritization, we use both Hidden Markov Model and an external morphological analyser to achieve high accuracy as well as high coverage. The morphological diacritization WER achieved by the system on the benchmark test set is 3.7%. We also introduce the use of Random Forest for the syntactic diacritization and show how this simple and light classifiers is very effective such that it outperforms very powerful classifiers by achieving syntactical WER of 8.3%. Finally, we provide time analysis for each component of the proposed system.

***Keywords:*** *Arabic Text Diacritization, Natural Language Processing, Machine Learning.*

## Nomenclature

| | |
|---|---|
| $BAMA$ | Buckwalter Arabic Morphological Analyzer |
| $CRF$ | Conditional Random Fields |
| $DER$ | Diacritic Error Rate |
| $DNN$ | Deep Neural Network |
| $HMM$ | Hidden Markov Model |
| $LDC$ | Linguistic Data Consortium |
| $NLP$ | Natural Language Processing |
| $OOV$ | Out of Vocabulary |
| $POS$ | Part of Speech |
| $SVM$ | Support Vector Machine |
| $WER$ | Word Error Rate |

## 1 Introduction

The Arabic language belongs to a class of languages that uses some subscript and superscript signs, called diacritics, to determine the exact pronunciation of words. Generally, different diacritics on the same letters produce different words with maybe very different meanings.
However, most modern standard Arabic scripts are written without any diacritics. We have some exceptions like important religious text, or text that is targeted to the beginner learners of the Arabic language. Given these facts, the lack of diacritics does introduce an additional source of ambiguity. Native Arabic speakers find it a very easy task to deduce most of the missing diacritics and to infer the correct pronunciation and meaning of the word using the information provided in the context. However, this is not an easy task for Arabic beginner learners and, of course, for NLP applications that need the input words to be presented in their diacritized form as a preprocessing step.
In his work [11], M.Maamouri et al. showed that NLP applications can benefit from diacritizing input text as a preprocessing step. They experimented the effect of diacritizing input text as a preprocessing step before handling the main task which was parsing. The results showed that the accuracy of the parsing is improved when the input text is diacritized. The improvement was not significant because both parsing and diacritization use very similar linguistic features. However, the effect of diacritization on other NLP applications is expected to be significant, according to Maamouri. Actually, a long list of NLP applications is in need of a reliable automatic diacritization system to achieve higher accuracy. Examples are Machine Translation,

Text To Speech, Automatic Speech Recognition, and Word Sense Disambiguation.

The Arabic orthography consists of 28 different letters; out of them, 25 letters are constants while the remaining three letters represent long vowels. Besides, 8 different diacritics are used to indicate the exact pronunciation of the letters. Table 1 lists different diacritics and the corresponding pronunciation resulting from attaching the diacritic to the Arabic constant letter "ر". Each of the Arabic letters can be diacritized using one or more diacritics. The only possible combination of diacritics that can be attached to a letter is: "Shadda" and one of these diacritics: "Fatha", "Kasra", or "Dumma".

Table 1: Arabic diacritic set

| Diacritic | Diacritized Letter | Pronunciation |
|---|---|---|
| Fatha | رَ | /r//a/ |
| Kasra | رِ | /r//i/ |
| Dumma | رُ | /r//u/ |
| Tanween Fath | راً | /r//an/ |
| Tanween Kasr | رٍ | /r//in/ |
| Tanween Dumm | رٌ | /r//un/ |
| Shadda | رّ | /r//r/ |
| Sukon | رْ | /r/ |

The process of Arabic text diacritization can be divided into two types: the morphological diacritization and the syntactic diacritization. To be fully diacritized, each word has to be diacritized both morphologically and syntactically. The morphological diacritization is the process of restoring the word diacritics that depend only on the word itself and its meaning. Morphological diacritics of a certain word, with a certain definition, stay unchanged regardless the context of the word. Actually, morphological diacritics distinguish a word from other words having the same letters. Table 2 shows an example of how different morphological diacritics on the same letters give different words with different meanings.

Generally, most Arabic native speakers predict the correct morphological diacritics very easily and at a very high accuracy using the information provided in the context.

Table 2: Examples of words with the same letters "عقد" but with different morphological diacritics.

| Word | Part of Speech | Meaning |
|---|---|---|
| عَقْد | Noun | Contract |
| عِقْد | Noun | Necklace |
| عَقَدَ | Verb | Held |
| عُقِدَ | Verb | Was held |

On the other hand, syntactic diacritics are dependent on the role of the word in the sentence. It can be inferred from the grammar of the Arabic language. However, the grammar controlling the syntactic diacritics consists of a large set of syntax rules. These syntax rules have many special cases such that even recent native Arabic speakers face considerable difficulty in predicting the syntactic diacritics correctly. Table 3 shows different syntactic diacritics of the same morphologically diacritized word.

Table 3: Different Syntactic Diacritics of The Same Word: "عَقْد"

| Sentence | Role of the word "عَقْد" |
|---|---|
| هَذَا عَقْدٌ بَاطِلٌ | Subject |
| وَقَّعْتُ عَقْدَ العَمَلِ | Object |

Actually, there are many challenges facing the problem of Arabic text diacritization. First, the nature of the Arabic morphological system itself is very complex. Generally, any Arabic word can be factorized into prefix, stem, and postfix. But, actually, there can be more then just one prefix and one postfix attached to a single stem. For example, the Arabic word "فسيكفيكهم" is itself a complete sentence which can be translated into English as: so he will suffice you against them.

The second challenge is the lack of large fully annotated corpus to be used for training. For example, the training corpus we use in this work, which is the benchmark training corpus, contains only 288.000 words. Of course, this number of words is not enough to train a system for applying such a sophisticated task at a high accuracy.

The third challenge is that for some Arabic words, different possible diacritizations can exist for the same word with exactly the same meaning and the same POS tag. For example the Arabic translation of the word "the prophet" can be "الرَّسُول" or "الرَّسُول". Another example is the Arabic translation of the word "electricity" which can be either "كَهْرُبَاء " or "كَهْرَبَاء".

Another challenge is that there is no unified way to write the Arabic sentence. The sentence can start with a word of almost any POS tag with no restrictions. The subject can come before or after the verb. Furthermore, the subject or the verb can be totally omitted from the sentence and left for the reader/audience to guess them. Finally, beside all these challenges that are inherent in the nature of the Arabic language, there are some very common spelling mistakes that exist even in formal documents; for example, letters in these sets (ا أ إ آ ), (ة ه), (ي ى) are commonly replaced, incorrectly, by other letters in the same set.

In this work, we extend our work in [12] by enhancing the operations of each of the system layers, introducing the use of the Random Forest classifier for syntactic diacritization , and embedding knowledge of the Arabic syntactic grammar to further improve the performance of the syntactic diacritization.


The rest of the paper is organized as follows: section 2 presents different approaches for handling the Arabic text diacritization problem, as proposed by other researchers. Then, we give the details of our proposed system, in section 3. Next, we describe the experiments we hold to test the performance of our system and to compare it against the state of the art systems, in section 4. After that, the time analysis is given in section 5. Finally, we give the conclusion and suggestion for future work in section 5.

## 2 Related Work

Researchers used several techniques for handling the problem of restoring missing diacritics. Some of these techniques are rule-based, morphological-based, statistical-based, or example-based. Regardless the variations of the details, these different techniques can fit into four main categories: (1) Dealing with the problem as a statistical machine translation problem. (2) Dealing with the problem as a sequence labelling problem. (3) Factorizing words into their basic segments using morphological analysis. (4) Hybrid of two or more of the approaches listed above. The text diacritization problem can be regarded as a statistical Machine Translation problem in which the undiacritized text is considered to be the source language text, and the diacritized text is considered to be the target language text. Elshafei et al. [7] proposed the use of HMM to predict the most probable sequence of diacritized words. The main disadvantage of their system is being of low coverage. The system fails to diacritize words that haven't been encountered in the training data. To face the problem of low coverage, Schlippe et al. [22] proposed a system that uses character level n-grams for OOV words. During the real operation of the system, every word is checked to determine whether it have been encountered during the training phase. If so, the word level n-gram model is used to determine the most probable diacritized word. Otherwise, word characters are diacritized using the character level n-gram model by choosing the most likely diacritized character given the neighboring characters.

Other researchers handled the problem as a sequence labelling problem in which every word is represented as a sequence of letters. Every letter may be tagged with any of the possible diacritics. Rashwan et al. (2014)[17] and Rashwan et al.(2015)[16] used a Deep Neural Network (DNN) based framework to restore the missing diacritics. They used a set of morphological, syntactic, and context features as input to the DNN. As a post-processing step, they introduced the use of Contention Sub-set Resolution network to enhance the performance of the syntactic diacritization. Although the deep neural network achieved very high accuracy compared to other proposed systems, the sequence labelling approach usually gives lower accuracy than other approaches. The exception is when the classifier is very powerful, like the DNN. But, the cost then is the large memory and time requirements. Zitouni et al. [25] used an approach based on maximum entropy to restore the missing diacritics. They used lexical, segment-based, and POS features for the maximum entropy classifier to label the characters with the missing diacritics. Instead of labeling each character independently, the labeling is chosen such that the conditional probability of the sequence of labeled characters is maximized. A beam search algorithm was also employed to reduce the search space.

In their system, MADA, Habash and Rambow [10]used the Buckwalter Arabic Morphological Analyzer(BAMA) to produce all the possible analyses for each input word. Then, they used a set of ten trained SVM classifiers to predict the expected POS tag features of the correct analysis. Finally, They chose the analysis that best conforms to the predicted POS tag features.

Usually, the hybrid approach combines both the machine translation approach and either the sequence labelling approach or the morphological analyser based approach to achieve high accuracy as well as high coverage. Rashwan et al.(2009) [15] introduced a two-layer system in which the first layer is used to get the diacritics of previously seen words and the second layer handles the out-of-vocabulary (OOV) words. In the first layer, analyses of words that occurred during training are retrieved and A* search is used to infer the most likely sequence of diacritized words. Then OOV words are passed into a factorizing module, which consists of a morphological analyser and a POS tagger, to get their possible analyses.

Ananthakrishnan et al. [1] adopt both statistical and knowledge-based approaches to solve the diacritization problem. In training phase, they built a statistical language model using Treebank data corpus. During the diacritization, the system at first uses word-level trigram model to obtain the most probable diacritization. But, if the word is OOV, the system gets all possible morphological analyses of the word using the BAMA morphological analyzer, and then it uses character-level 4-gram model to rank the possible dicacritizations of the word.

In [12], we introduced a hybrid technique that uses HMM and morphological analyzer to restore the morphological diacritics. Then, word feature vectors are built using some morphological and lexical features. A previously trained CRF classifier is then used to predict the syntactic diacritics.

## 3 Proposed Approach

We propose an approach consisting of 3 layers, which is an extension to the work we proposed in [12], in which each layer takes a step towards the complete diacritization of the input sequence of words. We first


handle the morphological diacritization of the input sequence, then we handle the syntactic diacritization of the morphologically diacritized sequence. The first two layers tackles the morphological diacritization of the input sequence. While the first layer is responsible for morphologically diacritizing words that have occurred during the training phase using first-order Hidden Markov Model (HMM), the second layer handles the morphological diacritization of the OOV words by making use of an external Arabic Morphological Analyser. After that, the third layer is there to handle the syntactic diacritization of the morphologically diacritized sequence by making use of the Random Forest classifier. Figure 1 shows the components of our approach and the interactions between these components. The details of each of the three layers are given in the following subsections.

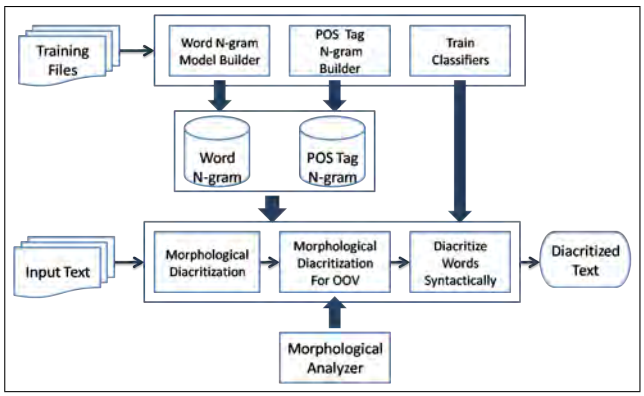



Figure 1: System Architecture

## 3.1 Morphological Diacritization of Previously Seen Words

Input words are fed into the first layer sentence by sentence. Each word in the input sentence is checked to determine whether it has been previously seen in the training data or not. If it's been previously seen, the different diacritizations of this word together with their unigrams are retrieved from the training database. If the word hasn't occurred during training, it is processed again in this layer as we begin to ask whether a variant of this word has occurred. We do so using simple letter normalization, excluding some parts of the word like prefix and postfix, and adding some of the allowed prefix and postfix to the word, then we check to see whether any of these variants of the word has occurred during training or not. If any of them has occurred, its diacritized forms are retrieved and the corresponding POS tags are formed as a combination of POS tag of the retrieved words with the POS tag(s) of the prefix and/or the postfix. Otherwise, if none of the word and its variants is found, the word is left unchanged and marked as OOV.

Then, to select the best probable sequence of morphologically diacritized words, we retrieve word bigrams from the database and apply first order HMM. To avoid the problem of zero and undefined probabilities resulting from OOV words, we use add-delta smoothing with $\delta = 0.05$. Upon selecting the words' morphologically diacritized form, POS tag bigrams are used to select the corresponding POS tags for the morphologically diacritized words. The flowchart describing the operation of layer 1 is shown in figure 2.

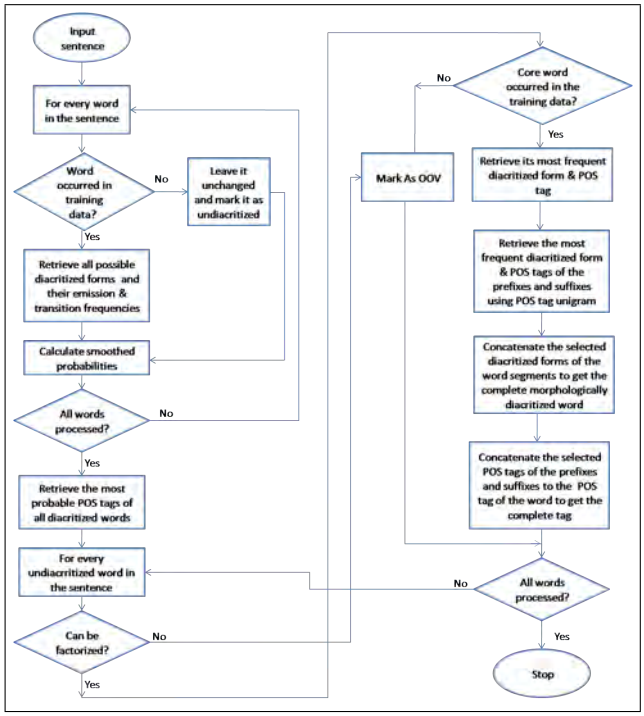



Figure 2: Flowchart of layer 1

## 3.2 Morphological Diacritization of OOV Words

The output of the first layer enters the second layer with the hope of getting the morphological diacritics for OOV words. As shown in figure 3, in this layer, we make use of the Buckwalter Arabic Morphological Analyser (BAMA) to get possible analyses for input words. The output of the analyser is a list of possible analyses for the input word. The analyses are sorted descendingly according to the frequency of the word. Each analysis consists of the diacritized form of the word and its corresponding POS tag.

Words are fed into the morphological analyser to get their analyses. If the word is previously diacritized in the first layer, we use the output of the morphological analyser only to extract some features. These features are: 1- Can the word be syntactically diacritized with any of the "Tanween" diacritics? 2- Is it common to omit the syntactic diacritic of the word?

On the other hand, if the word is OOV, the most prob-

able analysis is first chosen, and then these features are extracted. To select the most probable analysis, we make use of information gained during training about the expected POS tag as well as information provided by the morphological analyser about the relative frequency of different possible analyses. We used the following equation to calculate the probability of each analysis:

$$P(A_{i,j}) = \lambda.P(pos_{i,j}, pos_{j-1}) + (1-\lambda).\frac{1}{i+1}.N \quad )(1$$

Such that:

$A_{i,j}$ is the $i^{th}$ analysis in the output list of the morphological analyser for the word at position j, $pos_{i,j}$ is the POS tag of the analysis $A_{i,j}$, $pos_{j-1}$ is the selected POS tag of the word at position j-1, $\lambda$ is the linear interpolation parameter, $1 > \lambda > 0$, and N is a normalisation factor, $N = \frac{1}{\sum \frac{1}{i+1}}$

Once the analysis is selected, the corresponding diacritization and POS tag are assigned to the OOV word.

### 3.3 The Syntactic Diacritization

The task of the third layer, is to predict the syntactic diacritics of the input words. The input of this layer is the morphologically diacritized words that are annotated with their POS tags, as predicted by the two previous layers. Before we start diacritizing a word syntactically, we first check its POS tag. Using the POS tag of the word, we decide how the syntactic diacritization of this word should be handled. We classify every input word into one of three classes based on its annotated POS tag. The three classes are :

(1) Words annotated with noun, adjective, adverb, or number POS tags. (2) Words annotated with a POS tag of verbs in the present tense form used for singular subject. (3) Words of any other POS tag like prepositions, conjunctions, pronouns, verbs in the past tense form, and verbs used for plural subject.

We make this classification because the rules for diacritizing words in the same class are somewhat similar but different from the rules used for diacritizing words in other classes. Generally, words in the first class can be syntactically diacritized with any diacritic except "Shadda". But, the process of diacritizing these words is based on a large number of complex rules. Actually, the restoration of syntactic diacritics of words belonging to this class is much more complicated than the restoration of syntactic diacritics of words belonging to other classes. That's why, we use machine learning classifier to syntactically diacritize these words.

To predict the syntactic diacritics of words of class 1, we first need to build the feature vector. Beside the features extracted from the morphological analyzer in layer 2, some other features are extracted for each word to help predict its syntactic diacritic. We use a

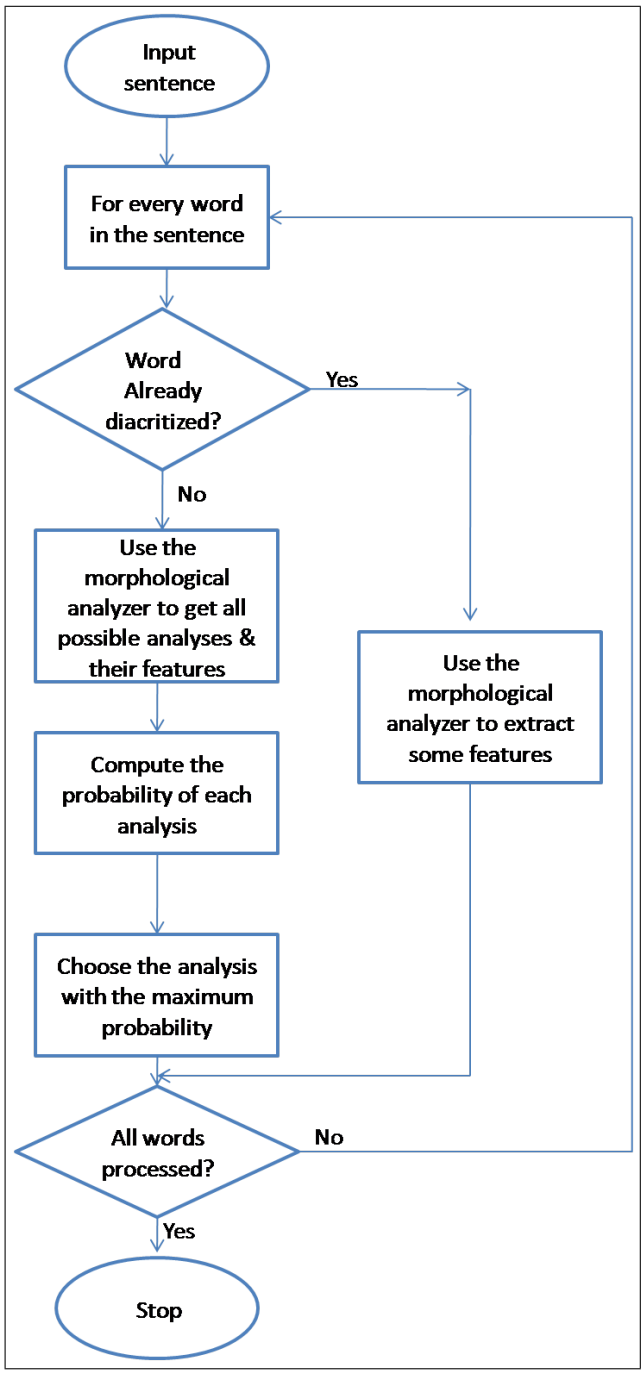



Figure 3: Flowchart of layer 2

combination of morphological, lexical, POS tag, and context features to build the feature vector. Then, we subject the feature vector to a previously trained Random Forest classifier. The Random Forest, we used, consists of 30 decision trees in which every decision tree votes for one of the candidate diacritics and the final decision, of the Random Forest as a whole, is taken as the diacritic with the highest number of votes. We used the implementation of the Random Forest that is provided in the open source "algLib" c# library [2].

The reason why we choose the Random Forest classifier for the syntactic diacritization task is that it is a rule based classifier in which each tree corresponds to a sequence of if-then-else checks. This conforms to the way in which people with strong Arabic grammar background predict the syntactic diacritics. Although people have the advantage that they make use of the semantics to further help them predict the syntactic diacritics at high accuracy, our technique still tries to mimic the way people perform this task but with much more limited sources of information.

As for words belonging to the second class, they can have any syntactic diacritic except "Shadda" and the three "Tanween" diacritics. There are a small set of easy rules that are used to select the correct syntactic diacritic for those words. Thus, words belonging to this class can be syntactically diacritized using a simple rule based classifier represented as a sequence of if-then-else statements. These if-then-else statements are directly inferred from the grammar that controls the syntactic diacritization of these words.

The syntactic diacritization of words belonging to the third class is a very trivial task. Actually, each of these words has a fixed case-ending diacritic which doesn't change in any context. So, the syntactic diacritic can be simply retrieved from the training database. The sequence of operations of layer 3 is summarized in figure 4. An example of how the system works on input sequence of words is shown in Figure 5.

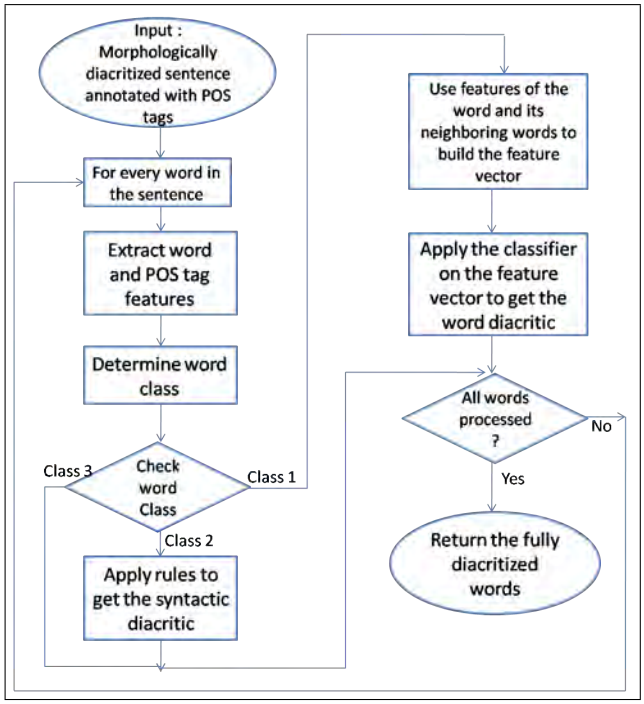



Figure 4: Flowchart of layer 3

# 4 Experimental Results

## 4.1 Data

We hold our experiments on LDC Arabic Treebank Part 3 which consists of about 600 articles from Al-Nahar Lebanese news agency. The words of the data set are annotated with their morphological and syntactic diacritics as well as with their POS tags. We use the same training and test data sets used by the state-of-the-art systems so as to guarantee fair comparison with them. The training dataset consists of 288K words representing articles from January 15, 2002 to October 15, 2002. On the other side, the test dataset consists of about 52K words representing articles from October 15, 2002 to December 15, 2002. Actually, training and test datasets were intentionally chosen to be chronologically non-overlapping so that the results model how the system will perform in the real world [20]. During the rest of the paper, we will refer to this test set as TestSet1.

Beside this benchmark test set of the Arabic Treebank, we also tested our system against another test set which is a collection of articles from "Aljazeera", annotated by RDI company in Egypt. The reason why we tested our system against this test set is that it's totally independent of the training data, as the articles are from different news agency and there is large chronological gap between this test data and the training data; most of the articles in this test data are published in 2012. This test data consists of 63 articles having about 30k words and 6k punctuation marks. The articles are of different topics from different categories. Their categories are: arts, economics, health, politics, sports, and varieties. We will call this test set TestSet2.

## 4.2 Results on TestSet1

Table 4 and table 5 show the comparison between our proposed approach and the state-of-the-art systems on TestSet1 in terms of the morphological diacritization WER and DER, the syntactic diacritization WER, and the complete diacritization WER.

The comparison indicates that both the approaches of Rashwan(2014) and Rashwan(2011) outperform our results in the morphological diacritization on TestSet1. However, we are still close to them, the difference is only about 0.7%. On the other hand, the

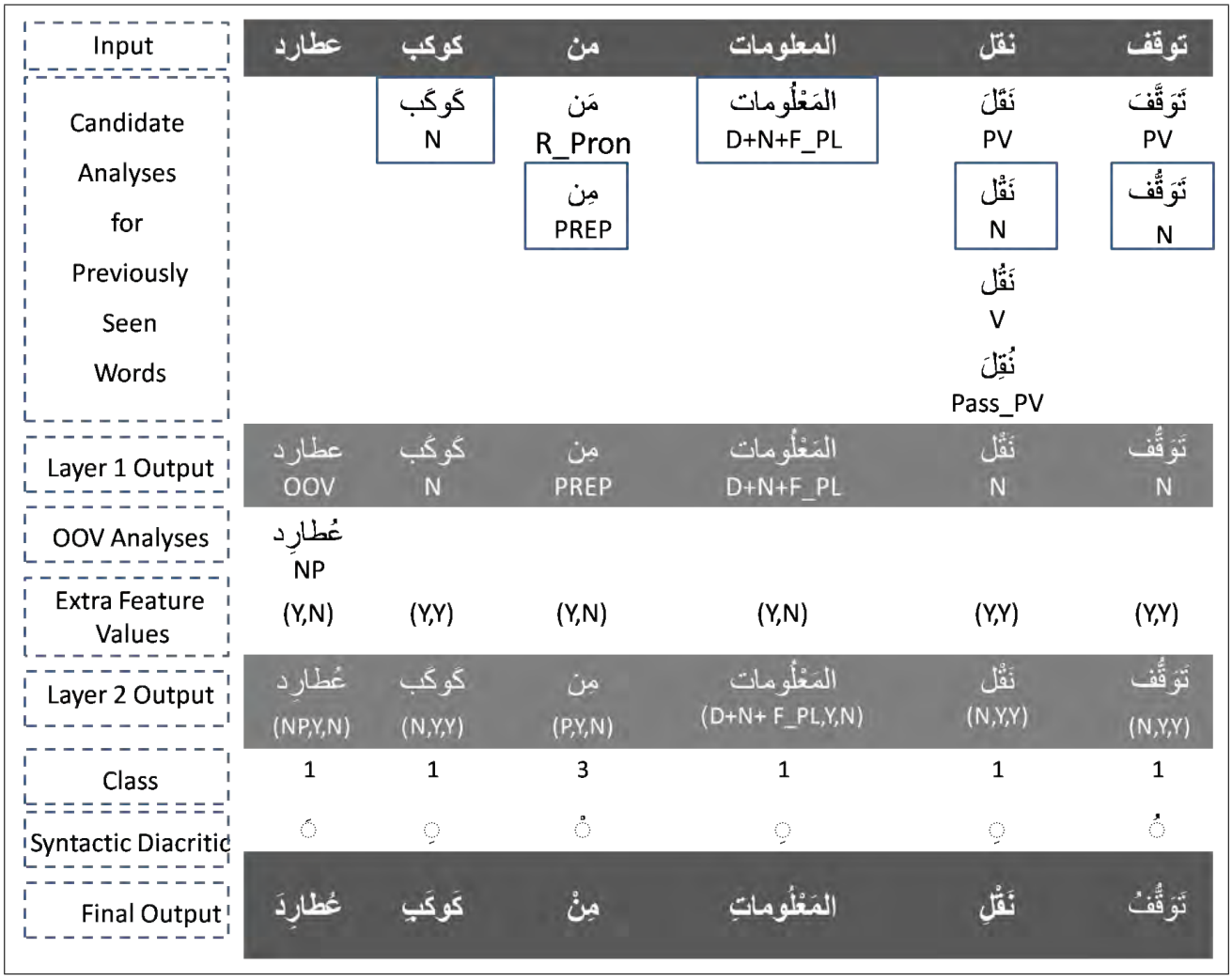



Figure 5: An example of applying the system on the input sequence "توقف نقل المعلومات من كوكب عطارد". In this figure, POS tags are abbreviated as follows: N stands for noun, R_Pron stands for relative pronoun, PREP stands for preposition, D stands for determinant, F stands for female, PL stands for plural, PV stands for past tense verv, V stands for present tense verb, and Pass stands for passive

Table 4: Comparison of the morphological diacritization WER and DER on TestSet1. The best reported results are written in bold

| Approach | Morphological WER | Morphological DER |
|---|---|---|
| Our Approach | 3.7% | 1.6% |
| Rashwan(2014) | **3%** | NA |
| Rashwan(2011) | 3.1% | **1.2%** |
| Zitouni | 7.9% | 2.5% |
| Habash | 5.5% | 2.2% |

Table 5: Comparison of the syntactic and full diacritization WER on TestSet1. The best reported results are written in bold

| Approach | Syntactic WER | Total WER |
|---|---|---|
| Our Approach | **8.3%** | 12% |
| Rashwan(2014) | 8.6% | **11.6%** |
| Rashwan(2011) | 9.11% | 12.5% |
| Zitouni | 10.1% | 18% |
| Habash | 9.4% | 14.9% |

comparison shows that our approach outperforms all the other techniques in terms of the syntactic WER by at least 0.3%. As for the total WER, the comparison shows that the work of Rashwan(2014), outperforms our results by only 0.4%. Figure 6 summarizes the results of our proposed approach in comparison with other approaches on TestSet1.

## 4.3 Syntactic Versus Case-Ending

The syntactic diacritic is the diacritic assigned to the last character of the core of the word, the stem, but not necessarily the last character of the word as a whole. On the other hand, the case-ending diacritic is the diacritic assigned to the last character of the word as a whole. For example, the word "يُعَلِمَهُ" has a syntactic diacritic "Fatha", but its case-ending diacritic "Dumma". If the word doesn't contain any postfix,

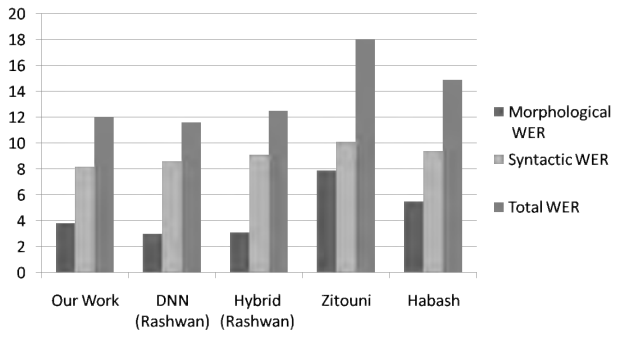



Figure 6: Graphical representation of the comparison between our system and the state-of-the-art systems on TestSet1

Table 6: The results of the system on TestSet1 using the concept of case-ending diacritic

| | |
|---|---|
| **Excluding last character WER** | 3.9% |
| **Case-ending WER** | 8.1% |

Table 7: The results of the system on TestSet2

| | |
|---|---|
| **Morphological WER** | 6.3% |
| **Syntactic WER** | 10.1% |
| **Complete WER** | 16.4% |

the syntactic diacritic and the case-ending diacritic are the same.

The syntactic WER is more accurate for expressing the powerful of the syntactic diacritization module than the case-ending WER. The syntactic WER is also exactly similar to the way people consider words correctly syntactically diacritized or not. Actually there is no real meaning for the case-ending WER in real life.

However, we noticed that some researchers reported the results of the case-ending WER instead of the syntactic WER. This also impacts the morphological WER, because the diacritic of a letter can be counted as a morphological diacritic or a syntactic diacritic( or case-ending diacritic) but not both. That's why we provide the results of our system using the concept of case-ending diacritic, here. The results of all word letters ignoring the last character diacritic as well as the results of the case-ending diacritization are reported in Table 6. Of course, the complete diacritization WER remains unchanged.

## 4.4 Results on TestSet2

As for the results of applying our system on TestSet2, they are reported in table 7.

The results show that the performance of the system is worse than its performance on TestSet1. We analysed the results and found that some words are counted as incorrect words while they should really be classified as correct by humans. This is because of the fact that some Arabic words have more than one possible diacritizations with exactly the same meaning and POS tag. 97 original Arabic words and 361 Arabic words of foreign origins exhibit this problem, in TestSet2 . We consulted the Arabic dictionaries to make sure that those original Arabic words are correct. The words "وَزارَة" , "وِزارَة" are examples of such words. As for the Arabic words from foreign origins like the words "أُغسطُس", "أَغسطُس" which can't be found in the Arabic dictionaries, we used our knowledge to decide that they are both correct. The results of the system on this test set taking into consideration the equivalent word diacritizations are reported in table 8.

Although this problem wasn't significant when we used TestSet1, it became significant when we tried

Table 8: The results of the system on TestSet2 taking equivalent diacritizations into consideration

| | |
|---|---|
| **Morphological WER** | 5.1% |
| **Syntactic WER** | 10% |
| **Total WER** | 15.3% |

this different test set because the articles in the extra test set are annotated by different human annotators with, maybe, different cultures or backgrounds.

# 5 Time Analysis

One main advantage of our system is its very competitive time requirements that make it a very powerful candidate for online operations. We will present time requirements, for both training phase and testing phase, in terms of the time complexity , in the following subsections.

## 5.1 Offline Operation Time Requirements

During the offline operation, we build the system, calculate all system parameters, and train system modules. Thus, the time complexity for training the whole system is the summation of time complexities of the three layers.

The time complexity of training the module of the first layer which uses first order HMM is the time for calculating unigram and bigram frequencies which is $O(N * logN)$; such that N is the number of words in the training data. For the second layer, no more parameters need to be calculated. Finally, to train the Random Forest in the third layer, the time complexity is $O(m * k * N * logN)$; such that m is the number of features and k is the number of decision trees in the Random Forest.
Thus,the time complexity of training the whole system is: $O(N * m * k * logN)$.

## 5.2 Online Operation Time Requirements

To calculate the total time complexity of the system in its online operation, we will sum up the time complexities of the three layer modules. The first layer uses first order HMM which has time complexity of $O(n * |S|^2)$; such that n is the length of the input sentence and $|S|$ is the number of different diacritizations, that occurred during training, for each word. In the second layer, the main operation is calculating the probability for each analysis so as to select the best probable analysis for a word form. Calculating the probabilities of h analyses of a word takes O(h) time complexity. So, the complexity of this module for input sentence of length n is $O(n * h)$. Finally, the last module uses Random Forest to get the most probable syntactic diacritic. Using Random Forest for a single word takes $O(k * log(N))$ where N is the number of training samples and k is the number of decision trees in the Random Forest. Hence the complexity of this module for an input sentence of length n is $O(n * k * log(N))$. Thus, the time complexity of the online operation of the system as a whole is: $O(n * (|S|^2 + h + k * log(N)))$. However, given the fact that S, h, k, and N are constants after the system has been built and trained, the overall complexity of the online operation becomes O(n).

# 6 Conclusion and Future Work

In this work, we proposed a multi-layer technique for handling the problem of Arabic Text Diacritization. The technique predicts both the missing morphological and syntactic diacritics at high accuracy. It uses first-order HMM as well as BAMA for morphological diacritization to achieve both high accuracy and high coverage. The WER of the morphological diacritization achieved by the system is 3.7% which is very competitive to the state-of-the-art techniques.
We introduced the use of the Random Forest classifier for the syntactic diacritization and showed that this very light classifier achieves very low syntactic WER, of just 8.3%. Our approach is much lighter than the best performing system, which uses Deep Neural Networks. However, the speed issues related to Deep Neural Networks make us a powerful rival for both online and offline operations.
As a future work, we may try to use a context sensitive morphological analysis component with the hope of improving the accuracy of OOV words.

# References

[1] Sankaranarayanan Ananthakrishnan, Shrikanth Narayanan, and Srinivas Bangalore. Automatic diacritization of arabic transcripts for automatic speech recognition. In *Proceedings of the 4th International Conference on Natural Language Processing*, pages 47–54, 2005.

[2] S Bochkanov and V Bystritsky. Alglib project.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automated methods for processing arabic text: from tokenization to base phrase chunking. *Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer/Springer*, 2007.


[5] Yousif A El-Imam. Phonetization of arabic: rules and algorithms. *Computer Speech & Language*, 18(4):339–373, 2004.

[6] Tarek A. El-Sadany and Mohamed A. Hashish. An arabic morphological system. *IBM Systems Journal*, 28(4):600–612, 1989.

[7] Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306, 2006.

[8] Raymond G Gordon Jr. Ethnologue: Languages of the world, dallas, tex.: Sil international. *Online version: http://www. ethnologue. com*, 2005.

[9] Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics, 2005.

[10] Nizar Habash and Owen Rambow. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics, 2007.

[11] Mohamed Maamouri, Ann Bies, and Seth Kulick. Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*, 2006.

[12] Aya S. Metwally, Mohsen A. Rashwan, and Amir F. Atiya. A multi-layered approach for arabic text diacritization. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 389–393, July 2016.

[13] Rani Nelken and Stuart M Shieber. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics, 2005.

[14] Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*, 2014.

[15] M Rashwan, Mohammad Al-Badrashiny, Mohamed Attia, and S Abdou. A hybrid system for automatic arabic diacritization. In *The 2nd International Conference on Arabic Language Resources and Tools*, 2009.

[16] Mohsen Rashwan, Ahmad Al Sallab, Hazem M Raafat, Ahmed Rafea, et al. Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):505–516, 2015.

[17] Mohsen AA Rashwan, Ahmad A Al Sallab, Hazem M Raafat, and Ahmed Rafea. Automatic arabic diacritics restoration based on deep nets. *ANLP 2014*, page 65, 2014.

[18] Mohamed Refaat, M Rashwan, and Nevin Darwish. Naïve bayesian modifications applied for automatic diacritization of arabic text. Master's thesis, Cairo University, 2011.

[19] Tim Schlippe, ThuyLinh Nguyen, and Stephan Vogel. Diacritization as a machine translation problem and as a sequence labeling problem. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA), Hawai'i, USA*, 2008.

[20] Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics, 2006.



# Biographies

**Aya S. M. Hussein** got her B.Sc. in Computer Engineering from Cairo University, Egypt in 2011. In 2015 she received her M.Sc. in Computer Engineering from Cairo University. Her research interests are mainly: Natural Language Processing, Machine Learning, Big Data, and Optimization.

**Prof. Mohsen Rashwan** received the B.Sc. and M.Sc. degrees in electronics and electrical communications from Cairo University, another M.Sc. degree in systems and computer engineering from Carleton University, Canada, and the Ph.D. degree in electronics and electrical communications from Queen's University, Canada. He currently serves as an Emeritus Professor in the Department of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, and as the R&D Director of RDI Arabic Language Technologies (HLT) research lab, (www.RDI-Eg.com). He has shared and led several national and international mega projects, like FP7 projects on Arabic HLT NEMLAR and MEDAR, and as a Co-PI in the Egyptian Data Mining Center of Excellence. He has cofounded ALTEC (an NGO to serve the Arabic language Technologies; www.ALTEC-Center.org).

**Prof. Amir F. Atiya** is currently Professor of Computer Engineering Department, Cairo University. He Was born in Cairo, Egypt. He received his B.S. degree in 1982 from Cairo University (Egypt), and the M.S. and Ph.D. degrees in 1986 and 1991 from Caltech, Pasadena, CA, all in electrical engineering. He held positions in academia, as well as several positions in financial firms. From 1997 to 2001 he was a Visiting Associate at Caltech. On leave from Cairo University, he recently held research positions in the firms Simplex Risk Management, Hong Kong, Countrywide Corporation in Los Angeles, and Dunn Capital Management, Florida. Currently, he is a Research Scientist with Veros Systems, Texas.



www.icgst.com

# Semantic Anti-patterns Detection in UML Models based on Ontology Catalogue

Eman K. Elsayed[1], Kamal A.El-Dahshan[2], Enas E. El-Sharawy[3], Naglaa E.Ghannam[4]
*Mathematical and Computer science Dept., Faculty of Science, Al-Azhar University, Cairo, Egypt*[1, 2, 3 &4]
emankaran10@azhar.edu.eg[1], dahshan@gmail.com[2], dr_enas_idm@hotmail.com[3], noga.yo99@yahoo.com[4]

## Abstract
To develop the pattern quality, we need to clear them from anti-patterns. This paper proposes a general semantic anti-patterns detection tool with an ontology catalogue of anti-patterns information retrieval. The proposed tool is able to identify the anti-pattern and its solution according to the error message. That is to detect the UML class diagram in design level. The system allows the user adding a new anti-pattern at certain category in the catalogue. That is, under the semantic condition to avoid redundancy**.**

***Keywords:*** *Ontology, Anti-patterns, Anti-patterns Catalogues.*

## Nomenclature

| Symbol | Meaning |
|---|---|
| UML | Unified Modelling Language |
| OOS | Object Oriented System |
| OWL | Web Ontology Language |
| RDF | Resource Description Framework |
| MOF | Meta Object Facility |
| SQL | Structured Query Language |
| XML | Extensible Markup Language |
| FOAF | Friend of a Friend |
| DCMI | Dublin Core Metadata Initiative |
| FQL | First Order Logic |
| EA | Enterprise Architecture |
| OLED | Ontology Lightweight Editor |
| OCL | Object Constraint Language |
| VPML | Visual Pattern Modeling Language |

## 1. Introduction
An anti-pattern describes a solution to a problem that generates negative consequences. Each anti-pattern has a name, type, description and solution. They appear in different levels, may be on the code level, design level or in implementation and every level has many numbers of the anti-patterns appear in it. Anti-patterns may be structure anti-patterns, behavioral anti-patterns or semantic anti-patterns, we only concerned with semantic anti-patterns in the Unified Modeling Language (UML) class diagram models.

All types of anti-patterns affect the quality of systems, causing wrong results and making many problems. Detecting anti-patterns is a good step to improve the quality of the models, especially in the design level; it is early more than code level that is considered too late. In addition, in the last years, the number of the anti-patterns became big and it is hard to remember all; there is no general catalogue that collects all types of anti-patterns with solutions of them.

There were many works of anti-patterns detection, as in references [1], [2], [3] and [4] but some of them just detected the inconsistency in UML class diagram and some detected the anti-patterns using different tool but without automatic solutions to the anti-patterns.

In addition, there were many works in the creation of anti-patterns catalogues, as in [5], [6], and [7]. These references created many catalogues, but they accepted the redundancy, the catalogues have no solutions and it covered only certain anti-patterns category. The proposed catalogue covered all anti-patterns categories, gave solutions and did not accept the redundancies. That is by using OWL ontology based to create the proposed catalogue.

In addition, we use the ontology to detect the semantic anti-patterns. An ontology is defined as a formal, explicit specification of a shared conceptualization [8]. An ontology consists of O = {C, P, R, T, I, A}. Where C is a set of concepts or classes; P is a set of properties of the concepts; R is a set of relationships between concepts; T is a set of hierarchically relationships among concepts that is called taxonomy; I is a set of instances and A is a set of axioms.

The ontology provides knowledge about specific domains that the computers and developers can understand. The ontology has many features as semantic properties that will help us recognize semantic anti-patterns and will help to prevent the duplication of the anti-patterns. The semantics of the ontology helps to share information [9]. The important point is the feature of semantic, which is available in the ontology.

The remainder of this paper is organized as follows: Related work is presented in section 2. The anti-patterns detection by ontology is presented in section 3. Then section 4 presented the empirical detection of the semantic anti-patterns. We introduce anti-patterns catalogue in section 5. Section 6 introduces the anti-pattern detection, correction, analysis and elimination. Finally, we present conclusion and future work in section 7.

## 2. Related Work

There were many works in the anti-patterns detection methods such as:

According to [1], the author detected the anti-patterns in MOF and UML meta-models by using the QVTRelation (QVTR) transformation, which distinguishes between two or more MOF-based models. A pattern is modeled with a Visual Pattern Modeling Language (VPML). This method detected the anti-Patterns through this transformation, but without any automatic suggestions to correct the anti-patterns. Although, this meta-model method have satisfactory performance, but our proposed method gives automatically the number of appearances, the name of these anti-patterns, the way to correct these anti-patterns through OCL constraints, and also detects the UML inconsistency.

In [2], they used an OntoUml editor to detect semantic anti-patterns in OWL but our proposed method is more general, it detects OntoUml anti-patterns in addition to other anti-patterns.

In [3], they presented the inconsistencies detection method in UML class diagram by using Net Beans 7.2.1 IDE. Our proposed method has a simple general way to detect the inconsistencies in addition to other anti-patterns.

In reference [4], the author detected the structure anti-patterns in UML class diagram by creating event-B model and solving the problems. Our proposed method detects the semantic anti-patterns using Ontology and makes a catalogue of all anti-patterns.

In addition, many anti-patterns catalogues were created by different tools and for different reasons.

In reference [5] the author presented an anti-patterns catalogue, this catalogue was just for showing meta-modeling anti-patterns, he described just UML anti-patterns, did not describe any other anti-patterns, giving us just the name of them and the query to detect each one. The catalogue didn't present any anti-pattern solution or the message appears in the case of it existed to be easy for the user to recognize it, so this catalogue was especially for showing meta-modeling anti-patterns not general, no solutions and no error messages. Our work is general as it shows all the anti-patterns and in the case of the user has the error message and does not know the anti-pattern, he gets all information about the anti-pattern that caused this message in an automatic way.

In reference [6] the catalogue showed and described many types of the anti-patterns; it described everyone, its type, its solution and some examples of it. When the user wants to know information about an error message, he cannot find the related anti-patterns. In addition, this catalogue has anti-patterns with different names -as "Blob" & "God Class"- and the same descriptions. Our work saves the time for all this, the user just enters the error message, he gets the anti-pattern information, and in addition, our catalogue prevents the duplicating of the anti-patterns.

In [7] the catalogue presented a list of different anti-patterns, it just described them, but there was no solution to any one of them, and it was for a certain type. Our work is general, gives a description and a solution to every anti-pattern and gets the information of the anti-patterns from the error message versa.

## 3. Anti-patterns Detection using OWL Ontology Based

Current approaches for identification of anti-patterns operate either at the code level (for software re-engineering purposes) or at the design level (for design quality improvement purposes). Detecting anti-patterns at the design level allows the designer to anticipate the problems that could be generated by an implementation. Detection of anti-patterns at the code level is considered too late. Therefore, we are interested in anti-patterns detection at the design level. There are several metrics approaches to detect anti-patterns as (Coupling, cohesion, complexity and inheritance).

In this paper, we concentrate on the object-oriented design, where we propose a metric-based approach for anti-patterns detection on UML class diagram designs. The proposed method for detecting anti-patterns based on mathematical metric between UML model and other formal models.

We use an ontology to detect the semantic anti-patterns in UML models to get a pattern. As Ontology has many benefits where it

- Improves the quality of the design and software systems.
- Selects semantic access and guided discovery of knowledge in Knowledge engineering and Management.
- Prevents the duplication.
- Ontology has the reasoner that we can use to check ontology consistency that is helping to avoid some anti-patterns.
- Ontology has many plugins as prompt plugin that helps to merge ontologies and make information retrieval from one project after the merging or any other processes it does.
- Ontology is the backbone of the semantic, so almost any semantic information will be stored in the form of it and published in the form of one of its languages as RDF or OWL.


- Ontology can perform a specific purpose as "Thesaurus" in the field of information retrieval or a model represented as OWL in the field of linked-data or XML schema in the context of databases or in software development.

We mapped UML to OWL, as there exists a semantic correspondence between them, which allows the automatic translation or conversion from UML to OWL. In addition, Ontology and UML are both object oriented models, so they have similar meanings as in table1.

**Table 1.** Correspondence between UML and Ontology

| **UML** | **Ontology** |
|---|---|
| Package name | Namespace |
| Class | Class |
| Instance | Individual |
| Attributes | Properties |
| Multiplicity | Cardinality |
| Generalization | Subclass of (Hierarchy) |
| Association | Relation |
| OCL constraints | Axioms containing these rules |
| (Superclass, Subclass) | The hierarchy concept (Taxonomy) |

The first step when we want to create the ontology is identifying all classes in the domain. Concepts refer to classes that are the most important component in ontology, where a class is a set of individuals that share common properties. According to [9] OWL classifies everything in terms of semantics that is helping the machine reader to know if an individual is a member of a class and its object properties or data properties related to that OWL class or not. Ontology describes the concepts and the relationships that are important in a particular domain. That is to provide a vocabulary for the domain [10]. A Class may have subclasses, or maybe it is a subclass of another class, but all of them are subclasses of the class "Thing" that is the root class in owl. The class hierarchy is called taxonomy.

In addition, a class may have some instances (individuals) and may not, it is not necessary to have.

Ontology has three types of properties, which are

- Object properties: Are the relationship between one individual and another individual of two owl classes.
- Datatype properties: Are the relationship between instances of classes.
- Annotation properties: They are used to add information to classes, individuals, and the others type properties.

Figure 1 presents our proposed system from the beginning, when any user enters into our system; he does one of three. First, he may has UML model and want to detect the semantic anti-patterns in it, second he may has an error message and wants to know the anti-pattern description and solution, or finally he may has new anti-pattern and wants to add it to the catalogue. The output will be the detected anti-patterns for the first choice, or anti-pattern information for the second choice, or the inserted anti-pattern information for the third choice.

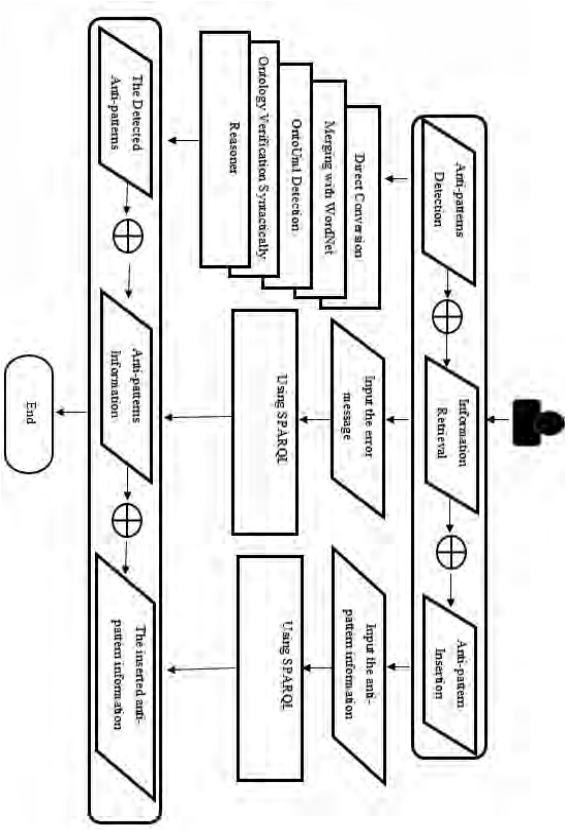



Figure 1. The proposed system

The following is the pseudo code for our system which applyes in more details in the following section.

*Input the choice*
***Case*** *(Anti-patterns Detection)*
*Insert UML model*
*Convert the UML to OWL*
*Run the detection processes*
*Print the detected anti-patterns list*
***Case*** *(Information Retrieval)*
*Input the error message*
*Run semantic search SPARQL query*
*Print the anti-pattern information*
***Case*** *(Inserting new anti-pattern information)*
*Run inserting SPARQL query*
*Output the new version of Anti-patterns ontology*
***End***

## 4. Empirically Detecting Semantic Anti-Patterns

We use the Protégé platform [11] which is an open source tool of ontology to do our work. We study the identification and repeating of these anti-patterns across different domains, different sizes and complexity.

We study, nine UML class diagrams which we uploaded them as UML templates to be used as patterns. These patterns are in (ATM UML class diagram [12], Library and Android UML class diagrams [13], Hasp UML class diagram [14], Seminar, Order and Auction UML class diagrams [15], SWT UML class diagram [16] and Furniture UML class diagram [17]).

To illustrate our work we use the example presented in Figure 2. The Hospital UML class diagram contains six classes "Hospital, Booking, Doctor, Patient, Continuous and not Continuous ". Class "Booking" has five attributes and three operations, which has one booking for everyone patient. Class "Doctor" has five attributes and no operations that will treat many patients. Class "Patient" has no attributes and four operations and the patient may be continuous or not continuous patient. Class "Hospital" has two attributes and no operations, class "Continuous" has two attributes and no operations and finally, class "not Continuous" has two attributes and no operations. The model also has seven associations, some of them with known multiplicity and some are not.

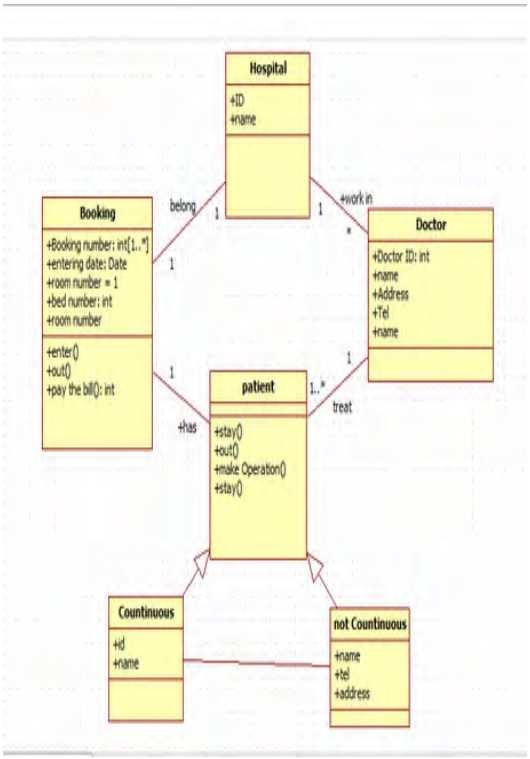



Figure 2. Hospital UML class diagram

For our example in Figure 2, when we transformed it to OWL directly:

- We detected seven anti-patterns, which are (Class has no operations, Class has no attributes, Class has no attributes and no operations, An attribute has no multiplicity, An attribute has no initial value, An operation has no return type and An association multiplicity is ambiguous). As classes "Doctor, Continuous and not Continuous" have no operations, class "patient" has no attributes; attribute "address" in class “Doctor” has no multiplicity. The same attribute "address" also has no initial value as in Figure 3. So Figure 3 shows both anti-patterns (Attribute has no multiplicity and attribute has no initial value). Class "Booking" has operation "Pay the bill" with a return type "Integer", but the operation "stay" in class "patient" has no return type , this anti-pattern is happening for all the rest of operations in class "Patient", and for operations "enter, out" in class "Booking". Finally, there are seven associations, three of them have associations with known multiplicity, but the association connects classes "Booking" and "Patient" has two ends, one with name "Has" has no multiplicity and another with multiplicity.

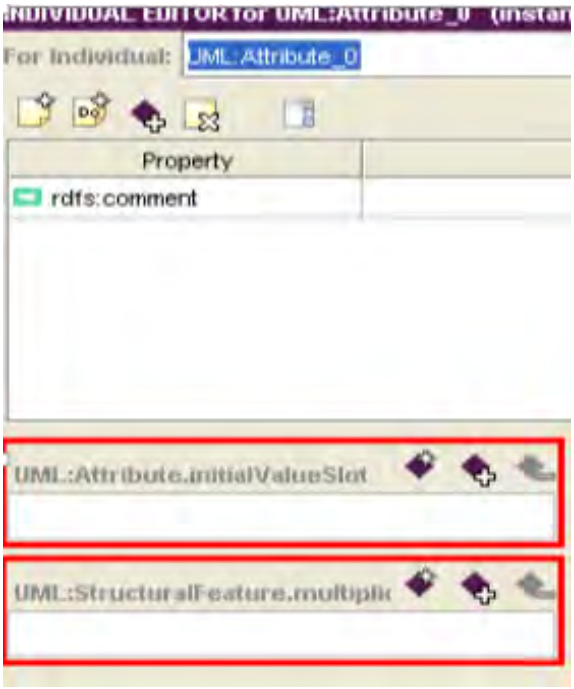



Figure 3. An attribute has no initial value and no multiplicity

- We detected "Same name" anti-pattern through merging the converted OWL ontology of UML with WordNet as used in [18], WordNet measures the similarity of meaning between two strings. But it is not enough. That is because there are semantic anti-patterns

were not detected by using WordNet or any linguistic ontology exactly in the UML model. In our example, the Hospital UML class diagram, this anti-pattern was detected three times, class (Booking) has attributes with the same name "room number", class "Doctor" has attribute "name" repeated twice and finally, class "patient" has an operation "stay" twice also.

- We also transformed UML to OWL not directly, by transforming the UML to OLED file in OntoUml editor, which is an Ontology Lightweight Editor (OLED) [19]. It imports the (Enterprise Architecture) EA file, which is created or imported from UML models. OntoUml editor has a list of twenty-one semantic anti-patterns giving us the name and the number of appearances. Some of them are explained in [2]. In our example, we detected three anti-patterns through using the validation of OWL anti-patterns in OntoUml editor. The three anti-patterns are "Association Cycle" as we have a cyclic relation among classes "Hospital, Patient, Booking and Doctor". The second anti-pattern is "Imprecise Abstraction" as we have a class "Patient" with upper multiplicity greater than one has two subtypes and it has a relation with the class "Doctor". The third anti-pattern is "Relation Composition" as there is an association between classes "not Continuous" and "Continuous" that is supposed to be disjoint classes as in Figure 4.

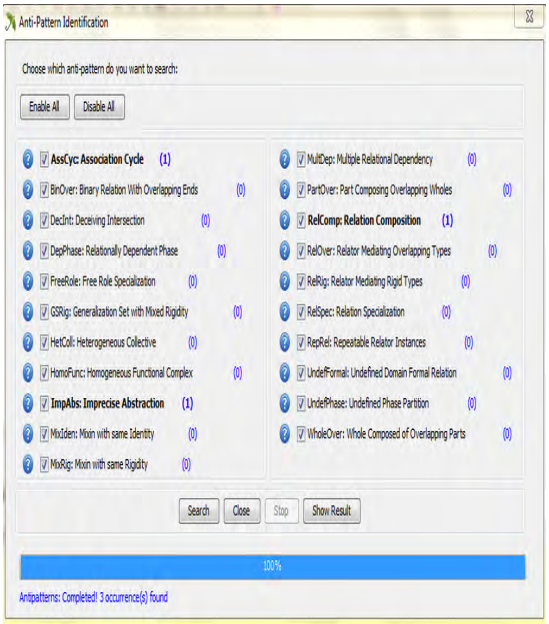



Figure 4. OntoUml semantic anti-patterns detection

In addition, by OntoUml we can detect the syntactical anti-patterns not just the semantic anti-patterns through using Ontology verification syntactically, it detects the anti-patterns (attribute has no data type, the class multiplicity equal zero and Invalid stereotype). In our example "Hospital" UML model, this verification detected the anti-patterns "Attribute has no data type" eight times, the anti-pattern "Class multiplicity equal zero "one time and "Invalid stereotype" nine times as in Figure 5.

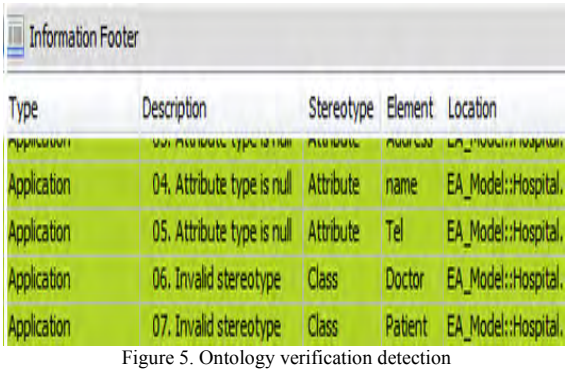



Figure 5. Ontology verification detection

- By the reasoner of ontology, we detect the inconsistency anti-patterns. This Reasoner detection can be done in the direct or not direct transformation. According to [3], the class diagram inconsistencies are (Similar name, Generalization and Disjoint, Multiplicity constraints and Cyclic Inheritance). ". After OntoUml validation and ontology verification, we transform the OLED file to OWL file. This transformation gives us the chance to detect the inconsistency using the "reasoner". In the "Hospital" UML model, "reasoner" detected three anti-patterns "same name", which we explained in the anti-patterns detected by merging with WordNet, "Cyclic Inheritance" which we explained in the anti-patterns detected by OntoUml, and "Generalization and Disjoints" which we explained in the anti-pattern detected by OntoUml with name "Relation Composition" between classes "Continuous and "not continuous".

Generally, according to this detection method, we detect thirty-six anti-patterns, which are thirty-three different semantic anti-patterns and three syntactical anti-patterns. As direct conversion detects seven anti-patterns, OntoUml has a list of 21 semantic anti-patterns, WordNet has one anti-pattern and Reasoner have four anti-patterns. All anti-patterns are in table 2, every detection tool has a true sign of its anti-patterns, some of the anti-patterns can be detected by more than one detection tool as Reasoner and OntoUml detect the anti-pattern "Association Cycle". And Reasoner and WordNet detect "Same name", so thirty-four anti-patterns will be just presented in table 2.

Table 2. The anti-patterns detected

| | The anti-pattern | Direct Conversion | OntoUml | WordNet | Reasoner | Proposed way |
|---|---|---|---|---|---|---|
| 1 | Class has no attributes | √ | | | | √ |
| 2 | Class has no operations | √ | | | | √ |
| 3 | Class has no attributes and no operations | √ | | | | √ |
| 4 | Attribute has no multiplicity | √ | | | | √ |
| 5 | Attribute has no Initial value | √ | | | | √ |
| 6 | Operation has no return type | √ | | | | √ |
| 7 | Association multiplicity is ambiguous | √ | | | | √ |
| 8 | Same name | | | √ | √ | √ |
| 9 | Generalization and Disjointness | | | | √ | √ |
| 10 | Multiplicity constraints | | | | √ | √ |
| 11 | Association Cycle | | √ | | √ | √ |
| 12 | Relation Specialization (RS) | | √ | | | √ |
| 13 | Relation Between Overlapping Subtypes (RBOS) | | √ | | | √ |
| 14 | Mixin with same Rigidity | | √ | | | √ |
| 15 | Binary relation with overlapping Ends | | √ | | | √ |
| 16 | Deceiving Intersection | | √ | | | √ |
| 17 | Relationally Dependent phase | | √ | | | √ |
| 18 | Free role specialization | | √ | | | √ |
| 19 | Generalization set with Mixed Rigidity | | √ | | | √ |
| 20 | Heterogeneous Collective | | √ | | | √ |
| 21 | Homogeneous Functional Complex | | √ | | | √ |
| 22 | Imprecise Abstraction | | √ | | | √ |
| 23 | Mixin with same Identity | | √ | | | √ |
| 24 | Multiple relational Dependency | | √ | | | √ |
| 25 | Part Composing overlapping whole | | √ | | | √ |
| 26 | Relation composition | | √ | | | √ |
| 27 | Relator Mediating Rigid Types | | √ | | | √ |
| 28 | Repeatable relator Instances | | √ | | | √ |
| 29 | Undefined Domain Formal Relation | | √ | | | √ |
| 30 | Undefined phase partition | | √ | | | √ |
| 31 | Whole Composed of Overlapping Parts | | √ | | | √ |
| 32 | Attribute has no datatype | | √ | | | √ |
| 33 | Class multiplicity equal zero | | √ | | | √ |
| 34 | Invalid Stereotype | | √ | | | √ |

In the study of the nine UML patterns plus our example, we divide the detected anti-patterns into four groups, which are (semantic anti-patterns of attributes, semantic anti-patterns of class, semantic anti-patterns of operations and semantic anti-patterns of association) as represented in table 3.

Table 3. Occurrences of Semantic Anti-patterns in the UML patterns

| **The anti-pattern group** | **% of occurrences across models** | **Total # of occurrences** |
|---|---|---|
| Semantic anti-patterns of attributes | %35.7 | 158 |
| Semantic anti-patterns of class | %42.1 | 186 |
| Semantic anti-patterns of operations | %5.4 | 24 |
| Semantic anti-patterns of association | %16.7 | 74 |
| **Total** | | 442 |

Figure 6 shows a snapshot of the system's output in the case of the detection process. The screen shows all anti-patterns were detected by our proposed method. But the detected anti-patterns in our "Hospital" example only have the true sign "√ " and if the user wants to retrieve the information about the detected anti-patterns, he just clicks the "Descripition and Solution" button. Figure 14 and Figure 15 display the example of the query output in this case

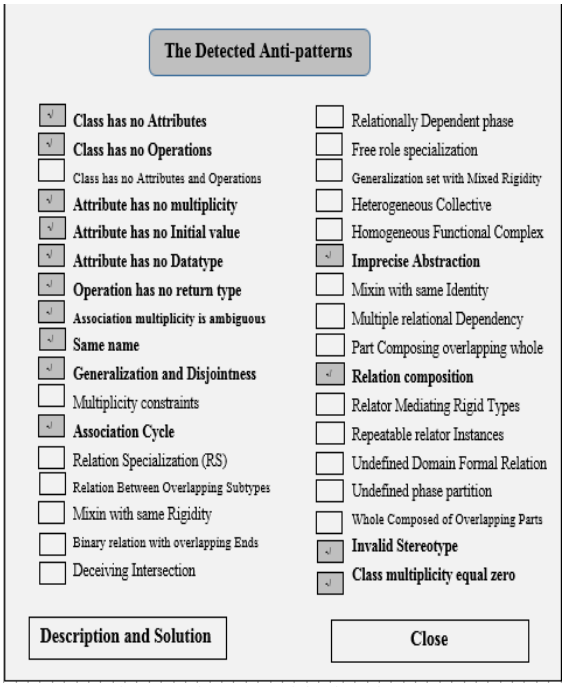



Figure 6. The output of the detection process

## 5. The proposed Catalogue of Anti patterns

There are many anti-patterns catalogues, but most of them just focuses on one special aspect. Many catalogues just present the anti-patterns that were detected by the work of any developer. In some cases, we cannot find a description of the anti-pattern, why it happened or the error message of it. We cannot find all information of the anti-pattern as they are distributed. We need an easy way to group and describe all the anti-patterns, helping to understand them to be avoided and to be easy for any user to get information that he needs. In our work saving the time, the money and getting correct results. Our proposed catalogue collects all the anti-patterns of different types such as design anti-patterns (semantic and structure), code anti-patterns, implementing anti-patterns, etc. As all the anti-patterns share the same domain that is the word (Anti-pattern) and all different types of the anti-patterns are concepts in this domain. Therefore, it was likely for us to use the ontology with its features specially the semantic feature for helping in semantic information retrieval.

We use the ontology editor protégé to create the catalogue. In addition, we can use the ontology reengineering tools as merging tool "as prompt" to

merge recent catalogues without duplicates. Then we use the reasoner to check the consistency of the catalogue.

We created the class "Anti-patterns" under the protégé root class "Thing". For every anti-pattern class, we specify its annotations, descriptions and its properties; we add possible error messages for every anti-patterns as individuals from different sources. According to [6] the anti-patterns may be in more than one category, which are (Software Architecture anti-patterns, Software Development anti-patterns, Software Management anti-patterns, User Interface anti-patterns and Organization anti-patterns). Every category has its own anti-patterns of different types and different detecting tools. The Software development anti-patterns were classified into (Design anti-patterns, Code anti-patterns, Scoping anti-patterns, UI anti-patterns… etc.) [20]. UML anti-patterns may be semantic or structure anti-patterns, also both of them has its detection tools, so in the catalogue, we create two classes, one of the semantic anti-patterns containing some tools that detect these anti-patterns as subclasses, each tool has the anti-patterns were detected by it. The second class "the structure anti-patterns" contains some detection tools and the anti-patterns that were detected by them. The anti-patterns ontology catalogue is shown in Figure 7.

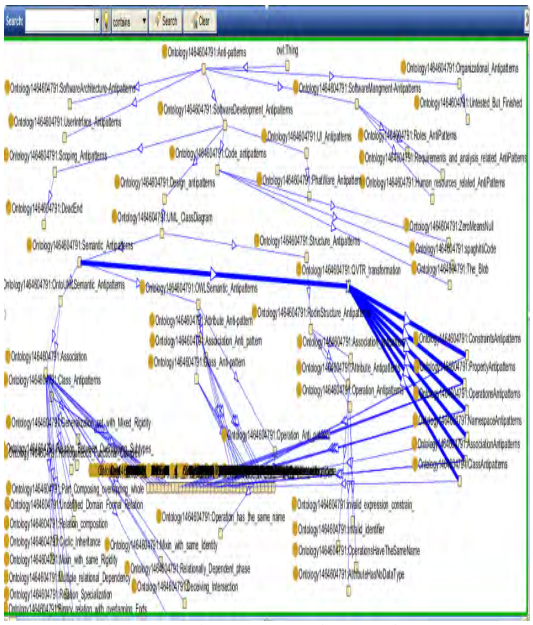


Figure7. Anti-patterns Ontology catalogue

This was for UML class diagram, but the same thing is also available for all types of UML diagrams, not just this, but also for any other modelling language, programming language like Java, C++, etc. This is just a case study, meaning that our catalogue is general; it has the ability to collect all anti-patterns. So anyone can add any type of anti-patterns to the catalogue, just adds all the information about it to make it easy for any other user to recognize it. First, we ensure that the anti-pattern is not already in the catalogue with a different name. In addition to the creation of the catalogue, we proposed a simple way to retrieve any anti-pattern information by using SPARQL. If the user has an error message, but he does not know the reason that causes this message, he simply enters the message to our system and all anti-pattern information will be the query result. This query result contains the name, type, detecting tool and solution to the anti-pattern. That is mean that all information are grouped in the same place.

Many users do not know many anti-patterns, and why the error message appears, almost they do not know the reason, so they just stop to search more and more in different sources of information, reading many anti-patterns descriptions, trying to reach for the anti-pattern to know the reason of this message. The proposed way helps them for saving the effort and the time by logging into the catalogue, writing the error message and in an automatic way, all information of the anti-pattern will appear. Moreover, in the case of the error message is not exist, the query result will be the anti-patterns names that their error messages have word(s) of the user message. Figure 8 presents the screenshot for structure anti-pattern that was detected by Rodin platform [21]; this anti-pattern is the "Invalid Identifier" anti-pattern. In addition, Figure 9 presents the SPARQL query of the anti-pattern; this query contains the complete error message not just a word. Figure 10 presents the query result.

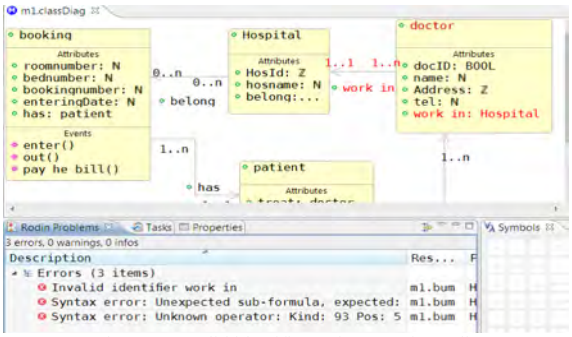



Figure.8. Invalid identifier anti-pattern in Rodin

In the case we have, the error message is not complete or is not the same according to the soft we use; the query result will be the more close anti-patterns to the message we enter. Figure 11 and Figure 12 are two anti-patterns that have a shared word as "attribute" in their error messages, when we query using this word; the anti-patterns that the error message of them contains this word will be the query result as in Figure 13.

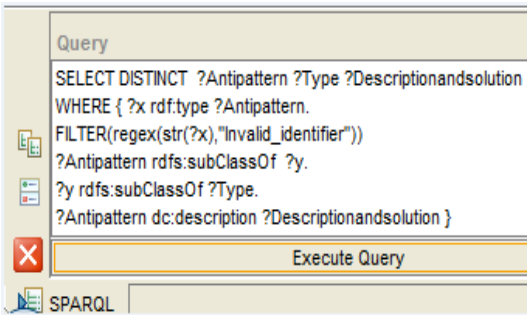



Figure9. Invalid Identifier query

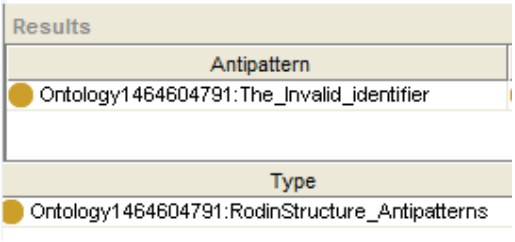



Figure 10. Invalid Identifier query result

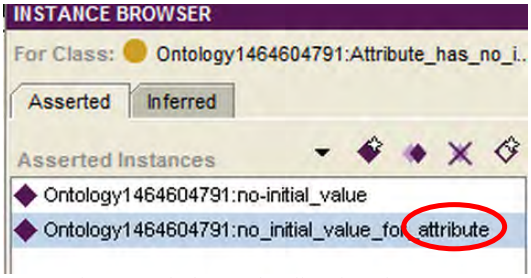



Figure 11. The keyword attribute in anti-pattern1

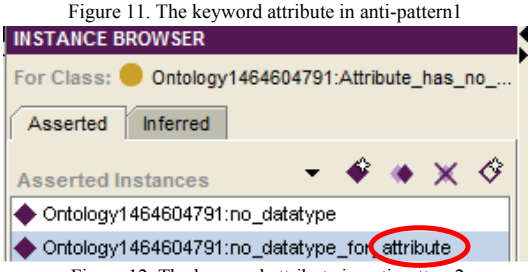



Figure 12. The keyword attribute in anti-pattern2

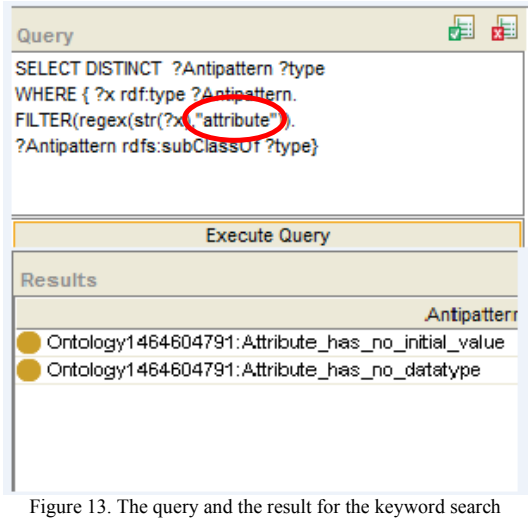



Figure 13. The query and the result for the keyword search

For the output of the detection process, when the user wants to know the description and the solution of the anti-patterns, the system retrieves the information that's stored in the catalogue using the query in Figure 14 and its result in Figure 15.

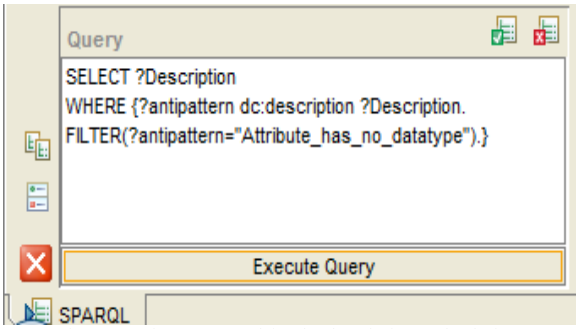



Figure14. The query used for the description and solution

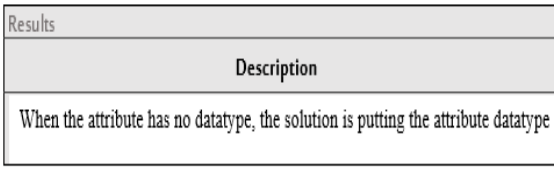



Figure15. The query result

Figure 16 shows the snapshot for the anti-pattern insertion in the catalogue which is for the user to enter the anti-pattern information.

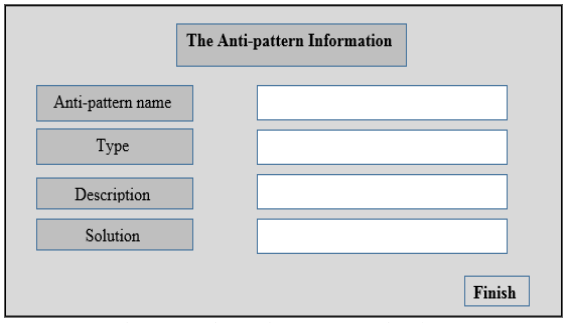



Figure 16. The Anti-pattern Insertion form

## 6. Anti-Patterns Detection, Correction, Analysis and Elimination

In this research work we use more than one detection tool, one of them was OntoUml that allows the automatic detection of anti-patterns and offers semiautomatic correction through the automatic generation of OCL constraints as in Figure 17 and Figure 18. The anti-pattern "Relation Composition" in the association between the classes "Patient" and "Doctor" is corrected by choosing the refactor option through answering some questions, after that the OCL constraints are generated to correct the anti-patterns in an automatic way.

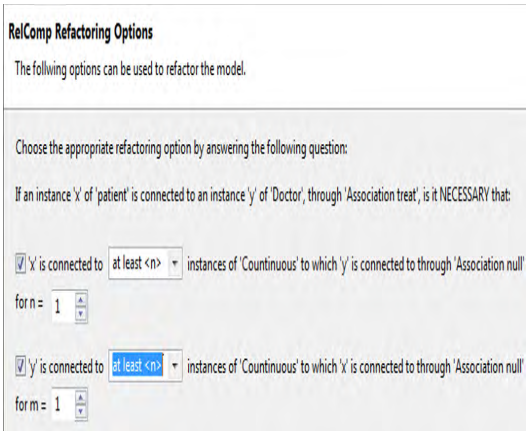



Figure 17. Choosing the refactor option

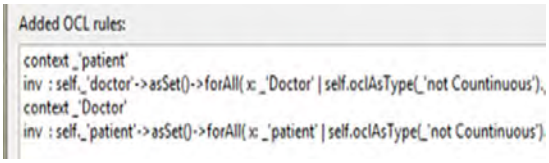



Figure 18. Automatically generated OCL solutions to the anti-pattern

In the direct conversion between UML and Ontology, we detect some anti-patterns that can be corrected through putting the missed part in Ontology directly, and then the anti-pattern will be corrected as in Figure 19. We correct the anti-pattern "Attribute has no initial value" through the sign "+", we add an initial value to the attribute in Ontology directly.


Figure 19. Correction in the direct conversion

For the "same name" anti-pattern which we detect by WordNet, the user just changes the name of the element.
For the inconsistency anti-patterns, the user sees the reasoner result and changes according to the result.
In the anti-pattern catalogue, we enter every anti-pattern with its solution into the catalogue. Therefor at the information retrieval process, each query result contains the anti-pattern associated with its solution.
The pseudo code for this step is as follows:
read the detected anti-patterns
IF(The anti-patterns are in the OntoUml list)
then
follow OCL constraints
Else IF (The anti-patterns are in direct conversion)
THEN
input the missing parts
Else IF (the anti-pattern is same name)
THEN
change the elements that have the same name
else if (The anti-patterns are inconsistency anti-patterns)
THEN
see the result of reasoner detection and correct
End

## 7. Conclusion and Future Work

The proposed method evaluates the quality of UML patterns. The two main distinctions that differ our proposed method from others are; first is that we made the integration between more than one detection tool, which are (ontology, WordNet, OntoUml and Reasoner) to detect the semantic anti-patterns, second, the correction of anti-patterns is automatic in some of anti-patterns
We applied the proposed method on a sample of ten UML class diagrams, which are uploaded as UML templates to be used as patterns in several online references. It detected thirteen semantic anti-patterns. In addition, it detected three syntactical anti-patterns. The "Semantic anti patterns of class" is the most commonly detected anti-patterns group, while the "Semantic anti-patterns of operations" is the least commonly appeared anti-patterns group. The total number of appeared anti-patterns in the sample of ten models is 442.
The main distinction that differ our proposed catalogue from others is the semantic information retrieval, which avoid redundancies in meaning.
In the future, we are going to make the anti-patterns fully automatic corrected. In addition, the highest result will be produced when we create a plugin for OWL in Rodin or create a plugin for Event-B in Protégé to detect both structure and semantic anti-patterns in one work. Finally, we will move to the next step by a general method that can detect both code and design anti-patterns levels.

## References

[1] M. E. Elaasar, L.C. Briand, Y. Labiche, PhD thesis: An Approach to Design Pattern and Anti-Pattern Detection in MOF-Based Modeling Languages. PhD thesis, Carleton University, (2012).
[2] G. Guizzardi, & Sales: Detection, Simulation and Elimination of Semantic Anti-patterns in Ontology-Driven Conceptual Models, Science. Vol. 8824, pp 363-376, Springer International Publishing, (2014).
[3] S.R. Idate and N.I. Dalvi. Method to Detect Inconsistencies from Class and Sequence Diagrams, International Journal of Science,

Engineering and Technology Research (IJSETR), Vol.3, Issue 8, (2014).

[4] E.K. Elsayed. Converting UML class diagram with anti-pattern problems into verified code relying on event-b. AIML journal, ISSN 1687-4846, Volume 14, issue 1, ICGST LLC, USA, (2014).

[5] Collection of anti-patterns available at https://sites.google.com/site/metamodelinganti-patterns/catalogueue. Access at (Oct. 2016).

[6] Collection of anti-patterns available at http://c2.com/cgi/wiki?Anti-patternsCatalogueue.Access at (Oct. 2016).

[7] Collection of anti-patterns available at https://en.wikipedia.org/wiki/Anti-pattern.Access at (Oct. 2016).

[8] N. Guarino, and R. Poli. Toward principles for the design of ontologies used for knowledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, in press. A substantial revision of paper presented at the International Workshop on Formal Ontology, (1993).

[9] Semagix.com. Advantage of Semantic data, available at http://www.semagix.com/advantage-of-semantic-data.htm, (2009).

[10] X. Jiang & A. H. Tan. Learning and inferencing in user ontology for personalized Semantic Web search, Information sciences, 179 (16), 2794-2808, (2009).

[11] Protégé Platform available at http://protege.stanford.edu/, (2015).

[12] ATM class diagram available at https://www.lucidchart.com/pages/class-diagram-for-ATM-system-UML, (2016).

[13] Library and Android class diagrams available at http://www.uml-diagrams.org/class-diagrams-overview.html, (2016).

[14] Hasp class diagram available at http://www.uml-diagrams.org/software-licensing-class-diagram-example.html?context=cls-examples, (2016).

[15] Seminar, Order and Auction class diagrams available at http://creately.com/diagram/example/gsxncbybt/Seminar+Class+Diagram, (2016).

[16] SWT class diagram available at http://www.ibm.com/developerworks/library/wa-eclipsemvc/, (2016).

[17] Furniture class diagram available at https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693, (2016).

[18] R. Fourati., N. Bouassida, and H. B. Abdallah. A metric-based approach for anti-pattern detection in uml designs". In Computer and Information Science, (pp. 17-33). Springer, Berlin, Heidelberg, (2011).

[19] OntoUML lightweight editor available at https://code.google.com/p/ontouml-lightweight-editor/, (2016).

[20] Development Anti pattern Road Map available at http://c2.com/cgi/wiki?DevelopmentAntiPatternRoadMap.

[21] Rodin platform available at https://sourceforge.net/projects/Rodin-b-harp/files/Plugin_%20UML-B.



## Biographies

Kamal Abdelraouf ElDahshan is a professor of Computer Science and Information Systems at Al-Azhar University in Cairo, Egypt. An Egyptian national and graduate of Cairo University, he obtained his doctoral degree from the Universit de Technologie de Compigne in France , where he also taught for several years. During his extended stay in France, he also worked at the prestigious Institute National de Tlcommunications in Paris. Professor ElDahshan's extensive international research, teaching, and consulting experiences have spanned four continents and include academic institutions as well as government and private organizations. He taught at VirginiaTech as a visiting professor; he was a Consultant to the Egyptian Cabinet Information and Decision Support Center (IDSC); and he was a senior advisor to the Ministry of Education and Deputy Director of the National Technology Development Center. Professor ElDahshan is a professional Fellow on Open Educational Resources as recognized by the United States Department of State.

Asoc. Prof. Eman K. Elsayed, PhD computer science 2005, Alazhar university, Master of computer science, Cairo University 1999, Bachelor of Science, mathematics and computer science Department, Cairo University 1994. She Published thirty three papers until 2016 in data mining, Ontology engineering, e-learning, image processing and software engineering. I also published two books in formal methods and event B on Amazon database. I am a member in Egyptian mathematical society and Intelligent computer and information systems society.

Dr Enas E. El-Sharawy, Computer Science M.Sc 2011 and PHD 2014 from Al-Azhar University, She works as lecturer of computer science at Al-Azhar University. She published many papers until 2014 in Formal method, Rodin platform and UML and interested in database and ontology.

Naglaa. E. Ghannam, Bachelors of Science, mathematics and computer science Department, Al-Azhar University 2002. Currently, She is a master student and working at Al-Azher University.